UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Paquetización de app movil con CMS. Xamarin-Wordpress

Autor/es

CARLOS AGUDO POSTIGO

Director/es

JESÚS MARÍA ARANSAY AZOFRA

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2019-20

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Paquetización de aplicación móvil con CMS.

Realizado por:

Carlos Agudo Postigo

Tutelado por:

Jesús M.ª Aransay Azofra

Logroño, febrero, 2020

Resumen

El presente Trabajo Fin de Grado tiene como objetivo realizar una abstracción de una aplicación móvil ya existente que se realizó durante el periodo de prácticas en la empresa para el Ayuntamiento de Ezcaray, junto con la modificación correspondiente de su gestor de contenidos, una página web realizada en WordPress, así como la modificación de la conexión entre ambas partes.

El resultado final de este trabajo es un software que nos va a permitir crear aplicaciones móviles de una manera sencilla, reduciendo los tiempos de desarrollo, definiendo únicamente el propósito de su creación y creando sus datos en el gestor de datos.

Además, se documentará el proceso de creación de nuevas aplicaciones móviles y entornos web para lograr agilizar este proceso, permitiendo así desarrollos futuros dentro de la empresa. También se creará una aplicación móvil como prototipo final enfocado en la Universidad de La Rioja para ilustrar y demostrar todos los avances realizados durante el proyecto.

Palabras clave: aplicación móvil, aplicación web, API, abstracción, gestor de contenidos.

Abstract

The present Bachelor Thesis is aimed to make an abstraction of an existing mobile application that has been done during the internship in the company, for the Ezcaray town hall, with the corresponding modification of its content management system, a website made in WordPress, and with the modification of the connection between the two parts.

The final result of this work is a software that will allow us to easily create mobile applications, reducing the development time, defining only the purpose of its creation and creating its data in the data manager.

In addition, the process of creating new mobile applications and web environments will be documented to speed up this process, for the purpose of encourage future developments inside the company. A mobile application will also be created as a final prototype focused on Universidad de La Rioja, to illustrate and demonstrate all the made progress during the project.

Keywords: mobile application, web application, API, abstraction, content manager.

Índice

Índice de Tablas

Índice de Imágenes

Capítulo 1

Introducción

De un tiempo a esta parte, la forma de consumir información de la red por parte de los usuarios ha sufrido un cambio notable. Los smartphones o móviles cada vez se han hecho más populares, logrando convertirse en un complemento casi imprescindible para la vida cotidiana. La conexión a internet de estos dispositivos, con su mejora cualitativa respecto a años atrás y la facilidad de adquisición de estos por su drástica disminución de precios, no han sido sino grandes atractivos a la hora de su demanda masiva.

Esto ha hecho que los dispositivos móviles entren en nuestra vida de una manera que bien podría parecer terrorífica en primera instancia, pero que no ha hecho sino facilitarnos la vida. Hemos pasado de usar el móvil para llamar, a usarlo para comprar o solicitar instantáneamente cualquier producto o servicio que necesitemos en ese momento, a poder consultar inmediatamente el tiempo de hoy, a qué hora pasa el autobús, como llegar a un sitio en concreto y, sobre todo, a usar el móvil para estar constantemente conectados logrando que un canal que en un principio se entendía como asíncrono se haya convertido, como por arte de magia, en un medio síncrono en muchos casos.

Gracias a esta proliferación de dispositivos móviles y paralelamente a ella, se ha abierto un hueco en el mercado enorme, que muchas empresas, desde hace tiempo han comenzado a llenar, como son las aplicaciones móviles. Las aplicaciones móviles han adquirido una incontemplable normalización en cuestión de 10 años.

En este momento tenemos "apps", si se me permite la abreviación, para prácticamente todo. Tenemos apps para ocio, como bien puede ser YouTube, Twitch, Netflix… Hasta aplicaciones para uso laboral como pueden ser, por ejemplo, Trello, aplicaciones para fichar, etc.

Dentro de este mercado surgieron también las aplicaciones de "Smart city", aplicaciones que permiten a cualquier usuaria o usuario estar enterada de todo lo que pasa en su ciudad o en la ciudad que vaya a visitar. Hasta este momento únicamente las grandes ciudades disponen de una aplicación de esas características, ya que los desarrollos de software en lo que compete al desarrollo de aplicaciones móviles son desarrollos costosos en tiempo y dinero, haciendo que pocas localidades puedan permitírselo.

De estos lodos, mientras desarrollábamos una aplicación móvil en mi empresa como requerimiento para uno de estos grandes municipios, fuimos conscientes que, si conseguíamos reducir los tiempos de desarrollo, y como consecuencia el coste de crear nuevas aplicaciones, seríamos capaces de brindar la oportunidad a los municipios más modestos de disponer, por qué no, de una aplicación "Smart city" propia.

Antecedentes

Previo a la finalización del anterior curso académico, me dispuse a buscar una empresa en la que poder empezar a hacer las prácticas en verano. Encontré una empresa que necesitaba una persona para un proyecto y cuya proposición me pareció bastante atractiva. En consecuencia, empezó mi periodo de prácticas el 1 de Julio en la empresa "The Demanda Valley". En ese tiempo comencé a formarme y a trabajar en un proyecto que consistía en la realización de una aplicación móvil multiplataforma, la cual correspondía a una necesidad de la empresa de abordar uno de los requerimientos de la licitación del proyecto que se estuvo desarrollando.

Ese proyecto consistía en realizar una aplicación móvil "Smart city" para el municipio de Ezcaray, La Rioja. La aplicación, además, debía estar focalizada en mayor medida en las rutas de "trekking" más famosas del pueblo, conocidas como las "7 rutas de Ezcaray", rutas que tradicionalmente han unido las diferentes aldeas de Ezcaray. Adicionalmente el Ayuntamiento de Ezcaray demandó una página web de referencia, como portal online para promocionar las rutas anteriormente mencionadas.

La empresa planteó en ese momento que sería beneficioso aprovechar esa página web como gestor de contenidos para nutrir la aplicación móvil, de ese modo eligió realizar una página en WordPress y un gestor de contenidos, que actuara como api para poblar de datos la aplicación, como un plugin que se añadiría a la página web.

En el momento de empezar este Trabajo de Fin de Grado, la aplicación móvil se encontraba en un estado primigenio, era funcional, pero necesitaba mucho trabajo, sobre todo de la parte del "back". Había que hacerla más eficiente, unificando o creando, en la mayoría de los casos, servicios que controlaran toda la funcionalidad de una manera transversal.

La empresa, al plantearse este proyecto de mejora, fue consciente que únicamente había que cambiar o mejorar el código de la aplicación, sino que, además, debía de mejorarse su API[1] correspondiente. De esa manera, se me asignó como responsable último del proyecto, pasando la empresa, representada por el que había sido mi tutor de prácticas Jesús Navajas Briones, a ser el cliente de este proyecto, junto con el cliente real del Ayuntamiento de Ezcaray que se beneficiaría indirectamente de todas las mejoras que se realizarán por mí.

Mi empresa se planteó que sería beneficioso aprovechar este proceso de mejora para paquetizar la aplicación y la API para que pueda ser reutilizado para futuras licitaciones para cualquier municipio que lo solicite. Ese fue el germen de este trabajo y por donde empecé a trabajar.

Hice un análisis de en qué punto partía la aplicación móvil en primera instancia, ya que a pesar de que era el campo en el que había estado trabajando en el periodo de prácticas, debido al plazo de entrega tuvo que intervenir mi superior en el proyecto, y, por tanto, el código estaba enmarañado y había partes que no tenía tan controladas. Gracias a ese análisis conseguí extraer ciertos requisitos que tener en cuenta a realizar en este trabajo. Por su parte, el superior de mi empresa, como cliente, me impuso una serie de requisitos que debían de hacerse para el trabajo.

Como siguiente paso, intenté analizar en qué punto partía la aplicación web correspondiente, ya que esta había sido realizada por un compañero de trabajo. Él en mayor medida fue quien me impuso una serie de mejoras a realizar que pasarían a ser requisitos.

[1] API: Application Programming Interface

Actores

Autoría

El responsable último de la realización de este documento y del correspondiente Trabajo de Fin de Grado es Carlos Agudo Postigo.

Tutores

Este trabajo ha sido posible gracias a Jesús María Aransay Azofra, tutor de prácticas y académico responsable del Departamento de Matemáticas y Computación de la Universidad de La Rioja.

Clientes

"The Demanda Valley", mi actual empresa representada por mi tutor de prácticas y del Trabajo de Fin de Grado, Jesús Navajas, como cliente, aportando funcionalidad extra a los requerimientos iniciales. Ayuntamiento de Ezcaray como cliente del proyecto realizado en prácticas del que se parte en este trabajo.

Punto de Partida

En esta sección vamos a detallar en qué punto parte la aplicación móvil y la API para así poder hacer un desglose y un mejor análisis de las tareas a realizar. Se ha realizado una ilustración para representar la arquitectura que tenemos ahora desplegada. El proyecto esta divido en dos partes claramente diferenciadas, la parte web y la parte móvil. (Ver Ilustración 1)

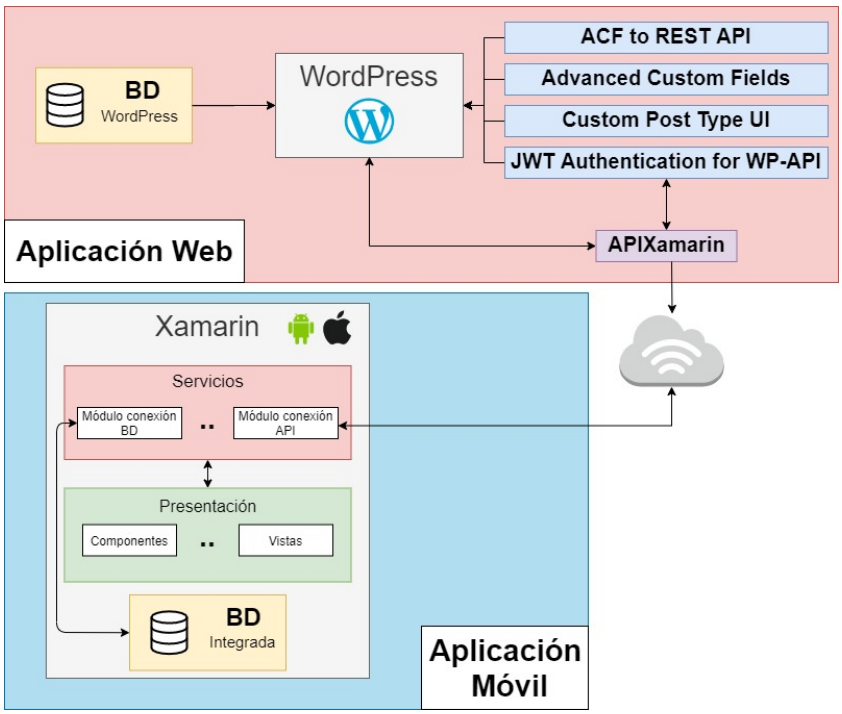


Ilustración 1 - Arquitectura

La parte web, como podemos ver en la imagen se centra en el editor de páginas web WordPress, con su base de datos propia y los plugin que utilizamos para brindar funcionalidad a la página web.

Dentro de la base de datos tenemos:

- **Archivos**: Son ficheros que se suben a la parte de media dentro de WordPress, pueden ser **pdf, videos, imágenes,** etc. Estos archivos serán referidos como **attachments**.

- **Datos**:

    - **Tipos de datos**: Estos tipos de datos serán referidos como **metadata.** Están creados con el propósito de clasificar los datos "data" (explicados en el siguiente punto) y se comportarán de la misma manera que los tipos de datos propios de WordPress, es decir, que todos los elementos de un tipo de dato tendrán los mismos atributos.

      Estos datos han sido creados debido a que no se podía enviar desde WordPress sus tipos de datos internos a la aplicación de una manera directa.

      Esto quiere decir que todos los datos que estén en WordPress seguirán la misma clasificación en la página web que en la aplicación móvil.

    - **Datos de contenido**: Estos tipos de datos serán referidos como **data**. Son elementos con determinados atributos propios que pertenecen a un "metadata" específico.

Dentro de la parte web tenemos los "plugins", que sirven para extender funcionalidad a la página web. Dentro de estos "plugins" tenemos:

- **Plugin de configuración de datos**: "ACF to Rest API", "Advanced Custom Fields" y "Custom Post Type UI".

- **Plugin de autentificación petición post (JWT Authentication for WP-API):** Este plugin se utiliza para asegurar que la petición post que se hace desde la aplicación móvil a la API para extraer todos los datos se haga de manera segura por medio de un "token". Este token está formado por los datos de autentificación de un usuario administrador habilitado para este propósito.

- **Plugin API (APIXamarin):** Este plugin desarrollado por mi empresa, particularmente para este proyecto, actúa como API para enviar toda la información que queremos a la aplicación y tener la misma información tanto en la página web como en la aplicación móvil. La API funciona brindando una "url" sobre la que realizar una petición POST incluyendo en su cabecera el "token" anteriormente mencionado. Esta petición post devuelve todos los datos en formato "json" para su lectura posterior en la aplicación móvil.

Por otro lado, tenemos la parte de la aplicación móvil. La aplicación móvil la estamos realizando en "Xamarin" ya que nos permite programar para "Android" y para "iOS" a la vez. Dentro del proyecto tenemos, como se puede ver en la imagen, una capa de servicios en la que tenemos por ejemplo un servicio que dice si está o no conectado a internet el usuario, un servicio de conexión y consulta sobre la base de datos, un servicio de conexión con la API, un servicio que gestiona el modelo de datos, etc.

También tenemos una capa de presentación donde encontramos tanto componentes propios como puede ser un control meteorológico, como controles para abrir páginas web dentro de la aplicación móvil, etc. En esta capa de presentación encontramos también las vistas y sus controladores. Asimismo, disponemos de una base de datos integrada que usamos a modo de caché donde almacenamos todos los datos que solicitamos en la petición.

Definición del sistema

Captura de requisitos

La captura de requisitos se hizo a través de reuniones con el tutor de la empresa. (Ver Anexo I, Acta de Reunión 1) Se determinó en relación con lo expuesto previamente en el punto "Punto de partida".

Requisitos Funcionales

Para la captura de requisitos se ha decidido dividir los requisitos funcionales en dos grupos, debido a que unos competen a la parte de la aplicación móvil, los cuales serán referidos con el código **"MR-XX" (Mobile Requirement)**, y otros competen a la parte del CMS, que serán referidos con el código **"AR-XX" (API Requirement)**. Asimismo, encontraremos requisitos que involucrarán a ambas partes los cuales se referirán como **"CR-XX" (Cross Requirement)** y requisitos que serán transversales a todos los "sprints" que serán referidos como **"CSR-XX" (Cross Sprint Requirement)**.

La tabla constará de tres columnas, una columna para el **código** del requisito con el que se referirán los requisitos dentro del presente documento, el **título** de este y una columna de **prioridad**, que hace referencia a que "sprint" corresponde, se explicará con más detalle en la sección "Metodología".

Resaltar que la captura de requisitos se realizará de manera aproximada, ya que, debido a la incertidumbre del proyecto en sí mismo no le es posible al cliente definir todos los requisitos en primera instancia. Debido a esto se utilizará una metodología ágil ya que nos ayudará a afrontar mejor cualquier imprevisto en lo que refiere a nuevos requisitos.

Requisitos Transversales (CSR-XX)

Código	Titulo	Prioridad
CSR-01	Ejemplo de creación de aplicación móvil sobre la Universidad de La Rioja a partir del paquete.	3

Tabla 1 - Requisitos Funcionales Transversales

CSR-01 - Ejemplo de creación de aplicación móvil sobre la Universidad de La Rioja a partir del paquete:

El objetivo de este requerimiento es introducir todos los datos necesarios para el entorno de pruebas y para el prototipo final. Así como la creación de páginas de contenidos para el prototipo final, como son formularios de incidencias, visores web, etc.

Requisitos Aplicación Web (AR-XX)

Código	Título	Prioridad
AR-01	Estandarización de Plugins.	0
AR-02	Mejora del rendimiento de carga de la API.	1
AR-03	Definir estándar para la estructura de datos de la API.	1
AR-04	Creación de menús dinámicos.	1
AR-05	Implantación de multidioma.	1
AR-06	Estandarización de pantallas de visualización/tipo.	1

Tabla 2 - Requisitos Funcionales Aplicación Web

AR-01 - Estandarización de Plugins:

Realización de una limpieza exhaustiva para el correcto funcionamiento del CMS[1]. La empresa en el proceso de realizar el CMS incluyó "plugins" de WordPress que no son necesarios o que bien pueden ser prescindibles. El objetivo de este requerimiento es dejar únicamente los plugin que aporten valor al CMS.

AR-02 - Mejora del rendimiento de carga de la API:

Reducción del tiempo necesario que precisa la API para generar los datos que serán enviados hacia la aplicación. El principal motivo de este requisito es el tiempo considerable que tarda en generar los ficheros de datos de la API.

AR-03 - Definir estándar para la estructura de datos de la API:

Definir como se representarán los datos para la comunicación API-APP. El objetivo es encontrar una solución debido a que la comunicación de la API con la aplicación es bastante caótica.

AR-04 - Creación de menús dinámicos:

Se buscará la mejor forma de abordar, diseñando e implementando una solución para lograr que los contenidos del menú lateral y del menú superior de la aplicación puedan ser definidos y actualizados sin necesidad de actualizar el código de la aplicación móvil, únicamente cambiando los datos en la API correspondiente.

AR-05 - Implantación de multidioma:

Se implantará la manera de adecuar los datos que se envían desde la API al idioma que se soliciten. El objetivo es añadir un parámetro más a la petición post a la API que incluya el idioma y devuelva los datos en su idioma correcto.

AR-06 Estandarización de pantallas de visualización/tipo:

El objetivo de este requerimiento es poder definir desde el CMS qué vista, del catálogo de vistas que dispone la aplicación, quiere el cliente que se utilice para cada tipo de dato. Tanto de su vista individual, como de su vista en lista, como el tipo de la lista en la que se mostrarán.

CMS[1]: Content Management System

Requisitos Aplicación Móvil (MR-XX)

Código	Título	Prioridad
MR-01	Reducción tiempo de carga de la aplicación.	2
MR-02	Mejora interacción aplicación móvil con su API.	2
MR-03	Estandarización de pantallas de visualización/tipo.	2
MR-04	Estandarización de estilos.	2
MR-05	Implantación de menús dinámicos.	2
MR-06	Implantación de multidioma.	2

Tabla 3 - Requisitos Funcionales Aplicación móvil.

MR-01 - Reducción tiempo de carga de la aplicación:

Requisito impuesto por el cliente, ya que en el punto en el que parte el proyecto la aplicación tarda en arrancar de media un minuto.

MR-02 - Mejora interacción aplicación móvil con su API:

Del mismo modo y análogamente al punto anterior se ha de mejorar el rendimiento y tiempos de espera en lo referente a la interacción APP-API.

MR-03 - Estandarización de pantallas de visualización/tipo:

De la misma manera que el requisito **AR-06,** el objetivo de este requerimiento es poder definir desde el CMS qué vista, del catálogo de vistas que dispone la aplicación, quiere el cliente que se utilice para cada tipo de dato. Tanto de su vista individual, como de su vista en lista, como el tipo de la lista en la que se mostrarán.

MR-04 - Estandarización de estilos:

Se buscará una solución para poder definir los ficheros de estilos de la aplicación móvil desde el gestor de contenidos.

MR-05 - Implantación de menús dinámicos:

Paralelamente al requisito **AR-04,** se realizará la arquitectura necesaria para generar tanto el menú lateral como el menú superior**,** cuyos datos recibirá heredados de su API correspondiente.

MR-06 - Implantación de multidioma:

De la misma manera que el requisito **AR-05**, el objetivo es que el dispositivo detecte el idioma empleado y solicite a la API los datos en ese idioma.

Requisitos Cruzados (CR-XX)

Código	Titulo	Prioridad
CR-01	Limpieza de código.	0
CR-02	Crear infraestructura de pruebas	0
CR-03	Conectar api-app.	0
CR-04	Creación entorno WordPress.	0

Tabla 4 - Requisitos funcionales cruzados.

CR-01 - Limpieza de código:

Este requisito surge debido a mi inexperiencia en cuanto al desarrollo de aplicaciones móviles, a la inexperiencia de mi compañero de empresa en el desarrollo de un plugin para WordPress y a la complejidad y el ruido que se genera en la plataforma de desarrollo elegida, Xamarin.

Además de que se han ido conservando partes de código que no tienen propósito alguno. Por consiguiente, el objetivo de este requisito es eliminar todas esas partes de código, dejando así, un código más limpio para su futuro mantenimiento.

CR-02 - Crear infraestructura de pruebas

Configuración de todos los "IDE[3]" necesarios para el desarrollo de este proyecto, así como un control de versiones como copia de seguridad y como repositorio para su futura descarga.

CR-03 - Conectar api-app

Integración de las dos partes para el desarrollo de este trabajo. El objetivo de este requerimiento es la creación unas credenciales con rol de administrador en WordPress, ya que, con esas credenciales se generará un token, una cadena de texto encriptada que contiene el nombre de usuario, contraseña y fecha de creación, que será incluida en la cabecera de cada petición que se haga a la api, con el propósito de asegurarnos de que solo aquellas personas que dispongan del token puedan solicitar los datos.

CR-04 - Creación entorno WordPress

Creación de servidor, base de datos, y entorno WordPress, elementos que junto al gestor de contenidos que actúa como "API" para la aplicación móvil hacen la aplicación web. El objetivo de este requisito es sentar las bases para la futura integración del gestor de contenidos.

[3]IDE: Integrated Development Environment

Requisitos No Funcionales

Del mismo modo que los requisitos funcionales voy a referirme a los requisitos no funcionales con el código **NFR-XX (Non-Functional Requirement).**

Código	Titulo
NFR-01	Entrega de Memoria el 19/02/2020.
NFR-02	Uso de PHP para desarrollo del plugin de WordPress.
NFR-03	Uso de Xamarin para desarrollo de App.
NFR-04	Uso de PHP para desarrollo del plugin de WordPress.

Tabla 5 - Requisitos no funcionales

A continuación, del mismo modo que los requisitos funcionales, se explicarán los requisitos no funcionales con más detalle:

NFR-01 - Entrega de Memoria el 19/02/2020

Requisito impuesto por la Universidad de La Rioja teniendo que depositar la memoria desde ese día hasta el 19/02/2020.

NFR-02 – Uso de WordPress para la creación del entorno web

Requisito impuesto por la empresa debido a la utilización de WordPress como entorno y como gestor de datos en el proyecto del que partimos.

NFR-03 - Uso de Xamarin para desarrollo de Aplicación móvil

Requisito impuesto debido a que la aplicación móvil, objeto de la reingeniería del proyecto, está desarrollada en "Xamarin".

NFR-04 - Uso de PHP para desarrollo del plugin de WordPress

Requisito impuesto debido al uso de un software cedido por un compañero, realizado en PHP al ser el lenguaje demandado por WordPress. Este software corresponde al plugin realizado previamente para el gestor de contenido de la página web.

Alcance

Dentro de los objetivos y requisitos de este proyecto se incluye completar los siguientes componentes:

- Como se ha estipulado en los requisitos se realizará una aplicación móvil en Android sobre un caso de prueba, probando la flexibilidad de la aplicación para adaptarse a distintos escenarios.
- Se descarta realizar la misma en iOS ya que, en este momento no se dispone de un ordenador "Mac" necesario para compilar la aplicación.
- Se descarta realizar una visualización o adaptación web de todos los datos (data), tipos (metadata) y ventanas que se utilicen para la aplicación móvil y en su gestor de datos para el prototipo final.
- La aplicación móvil y su web correspondiente no dispondrá de una gestión de usuarios, posponiendo esto para futuras iteraciones dentro del marco de la empresa.
- La aplicación y su demostración se harán en modo local, ya que la empresa no dispone de una infraestructura suficiente como es el caso de servidores propios, al ser una empresa pequeña.
- Todos los avances realizados en este trabajo deberían ser reutilizables por la empresa para la futura realización de aplicaciones móviles, cuando se considere necesario.
- Se realizará documentación de manera paralela a la realización del trabajo, con el objetivo de facilitar la configuración, instalación y funcionamiento para toda persona que lo necesite.

Exclusiones

Este trabajo **NO** incluirá las siguientes acciones:

- Mantenimiento de la aplicación en el tiempo.
- Despliegue de las aplicaciones en iOS y Android en sus "stores" correspondientes.
- Despliegue de página web funcional con su visualización.

Entregables

Como resultado de este trabajo resultarán los siguientes entregables, que serán divididos en entregables de software, referidos con el código **SP (Software Product),** y documentos, referidos con el código **DP (Document Product)**.

Documentos:

Código	Titulo
DP-01	Memoria del Trabajo
DP-02	Manual de uso negocio.
DP-03	Seguimiento de reuniones.

Tabla 6 - Entregables I

- **DP-01 - Memoria del Trabajo:**

  El presente documento, resultado del trabajo de fin de grado.

- **DP-02 - Manual de uso negocio:**

  Documento de explicación de cómo incluir nuevos datos en WordPress. (No se añadirá como anexo ya que es uso exclusivo de la empresa)

- **DP-03 - Seguimiento de reuniones.** (Incluido como anexo en el presente documento)

Software:

Código	Titulo
SP-01	Plugin parametrizado de WordPress.
SP-02	Aplicación de móvil parametrizada.
SP-03	Aplicación de pruebas.

Tabla 7 - Entregables II

- **SP-01 Plugin parametrizado de WordPress.**
- **SP-02 Aplicación de móvil parametrizada.**
- **SP-03 Aplicación de pruebas:**

  Aplicación de pruebas que será utilizada para demostrar todo el trabajo realizado en este proyecto.

Planificación

Metodología

Para este tipo de proyecto se ha decidido usar una metodología ágil. He decidido realizar una metodología ágil ya que en primera instancia ni yo ni la empresa como cliente somos capaces de definir todos los requisitos funcionales y no funcionales previo al desarrollo del proyecto. Además, el propósito último de este proyecto está condicionado por el trabajo previo que se ha realizado y por tanto no nos es posible realizar un desarrollo puramente en cascada. Por lo tanto, se ha decidido seguir una metodología de trabajo iterativa trabajando por medio de sprints.

En la parte de definición del sistema se definieron una serie de requisitos dotados de prioridad de manera que los requisitos de nivel 0 serán imprescindibles para poder realizar el resto en los siguientes sprints y, por tanto, los indicados a realizar en el primer Sprint. Estos requisitos corresponden a todo el análisis y desarrollo previo a la realización del proyecto, como por ejemplo la configuración de los sistemas.

En el primer sprint, que se realizará a continuación del Sprint 0, se desarrollarán los requisitos con prioridad 1, y los requisitos que se hayan quedado sin hacer en el Sprint 0, si es que hubiera alguno. En el sprint 2 se desarrollarán homólogamente los requisitos con prioridad 1 que hayan quedado sin hacer en el Sprint 1 y requisitos con prioridad 2. De la misma manera, en el Sprint 3 se desarrollarán los requisitos con prioridad 2 que hayan quedado sin hacer y requisitos con prioridad 3.

Pese a que los "sprints" se suele asumir que deben de tener el mismo tamaño en cuanto a tiempo, consideramos, y así ha sido reflejado en la Estimación Temporal y en el Diagrama Gantt correspondiente, que, por motivos de estimación, los "sprints" durarán la suma de las estimaciones de todos los requisitos que se van a desarrollar en ellos.

Además, pese a que a las partes del proceso se les han llamado "sprints" por seguir con la notación propia de la metodología y, sabiendo que un "sprint" en metodología ágil se puede denominar incremento ya que supone un incremento de la funcionalidad en el prototipo resultante, en nuestro caso, esa característica de incrementalidad puede que no sea reflejada en el prototipo final, pero todo sprint aportará valor y servirá como propósito para el fin último de este proyecto.

Al inicio de cada sprint se seleccionarán todos los requisitos a realizar en ese proyecto, denominando este proceso como Identificación. Posteriormente se irán realizando dentro del mismo sprint diferentes tareas y en cada tarea pueden realizarse uno, de manera general, o varios requisitos simultáneos. Esta parte del "sprint" se le denominará proceso y se subdividirá en tareas (de 0 a n). Todo requisito que no se haya podido realizar en el proceso será requisito del siguiente "sprint", por lo que, al final de cada sprint se realizará un "Review" en el cual se identificarán todos esos requisitos que no ha dado tiempo a realizarse y se analizará si durante el proceso de desarrollo del sprint se han encontrado nuevos requisitos y qué prioridad se les concede.

Por lo tanto, como se puede ver, no se está aplicando una metodología ágil al uso, sino más bien es una adaptación libre de la misma, pero, sí que se utilizarán herramientas y procesos de trabajo propias de este tipo de metodología, como las que serán explicadas en el siguiente párrafo.

Uno de los procesos o características que hemos creído necesarios incorporar de la metodología ágil es el contacto directo con el tutor de empresa, haciendo las veces de cliente, por medio de reuniones. Como se ha explicado anteriormente, el proyecto tiene bastante incertidumbre y creemos que es necesario estar, durante la realización de este, muy estrechamente relacionados con él. En nuestra casuística tan especial, el cliente es la misma persona que el tutor de la empresa, ya que este proyecto está destinado a la empresa para facilitar cualquier desarrollo posterior ante cualquier nueva licitación que se nos requiera, reduciendo así la complejidad y el tiempo necesario para llevarlo a cabo.

Para el contacto con el cliente se ha optado por el mantenimiento de una conversación tanto directa como indirecta. La comunicación directa se intentará realizar prácticamente todas las semanas, ya que estará sujeta a la disponibilidad del cliente, y debido a que el cliente reside en una ciudad diferente de la nuestra la opción que se ha elegido para esta empresa es la comunicación directa a través de "Skype".

Del mismo modo, para no atosigar al cliente con reuniones para cada aspecto relacionado con el desarrollo, se ha optado por mantener una comunicación análoga por medio de "Trello".

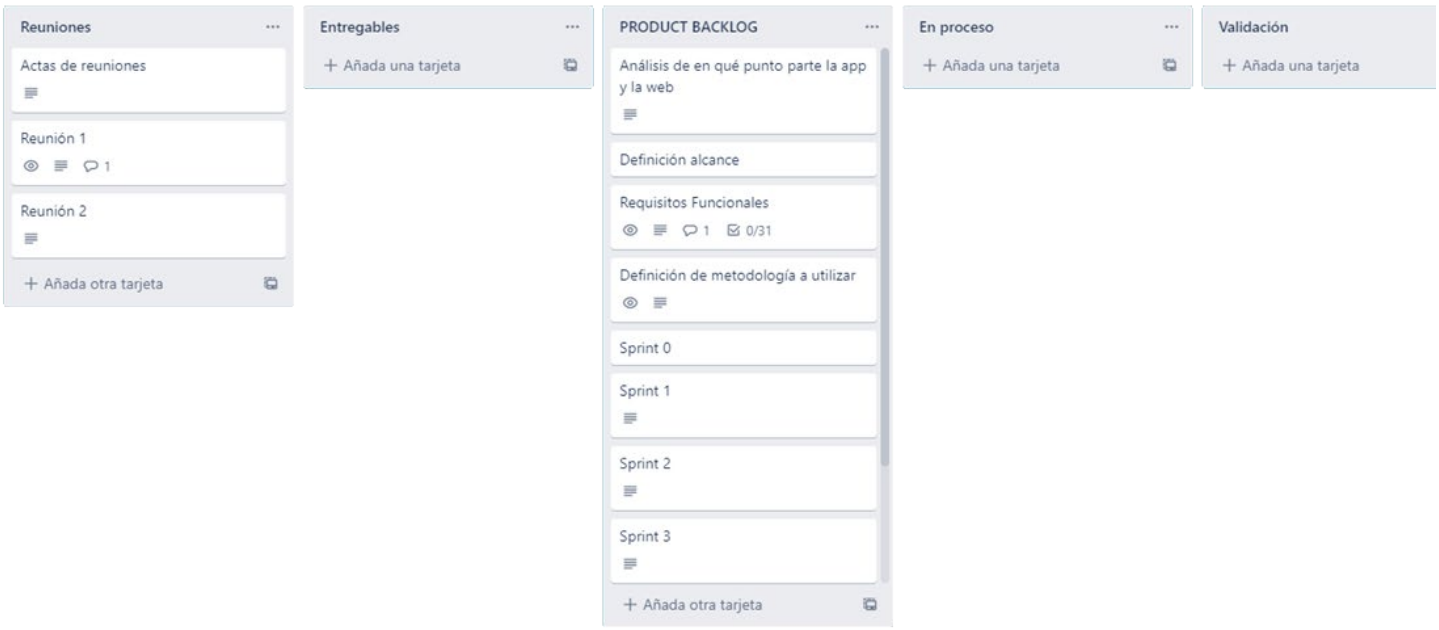


Ilustración 2 - Tablero de Trello

En el tablero (Ver Ilustración 2) podemos encontrar cinco columnas:

- **Reuniones:** Seguimiento de reuniones.
- **Product Backlog:** En esta columna se encontrarán todos los requisitos que se han definido en la parte de Definición de sistema, al inicio del proyecto, y todos los que posteriormente se irán añadiendo como consecuencia al desarrollo del proyecto.

- **En proceso:** Todos los requerimientos que se estén realizando en este momento. Hemos creído necesario la existencia de esta tabla para que el cliente sepa en todo momento qué se está realizando, pudiendo transmitirnos su inconformidad de los requisitos tratados en ese momento si considera que hemos de tratar otros con mayor prioridad, por cualquier motivo, a pesar de que él mismo ya ha definido una prioridad previa a la realización del proyecto para cada requerimiento y se ha realizado una planificación en consecuencia.

- **Validación**: Una vez se finalice un requerimiento se pasará a esta columna, dejando en las manos del cliente su validación, proporcionándole los medios necesarios para este propósito y pudiendo comentar en la misma tarjeta todo aquello con lo que no esté de acuerdo. Si no existiera ningún problema el requerimiento pasaría a la columna de "Finalizado".

- **Entregables**: Productos que serán el resultado del proyecto.

Dependencias identificadas

En la siguiente ilustración (Ver Ilustración 3) podemos ver las dependencias entre las diferentes partes del trabajo que encontraremos en el desarrollo del proyecto y que determinarán el proceso de planificación de este.

En este punto vemos la descomposición de tareas de los principales paquetes de trabajo que han sido identificados, esta distinción se hizo como complemento a la metodología. Gracias a este esquema se consiguieron asignar las prioridades con mejor precisión, ya que es importante recordar que hemos ido asignando prioridades a las tareas según su necesidad para poder desarrollar otras tareas del TFG.

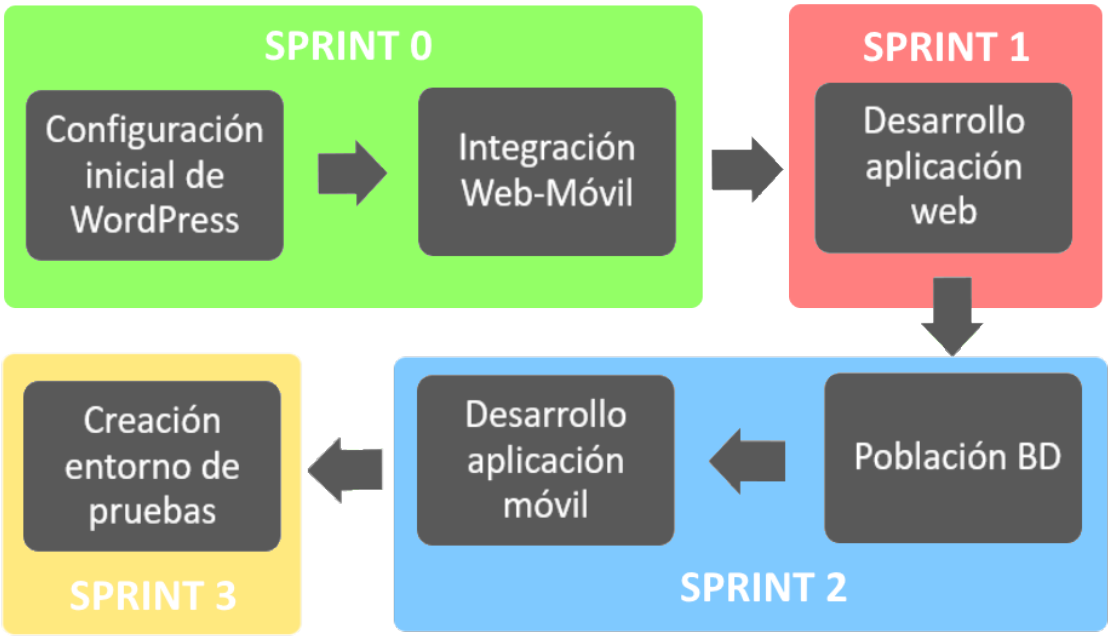


Ilustración 3 - Gráfico de Dependencias

Esquema de Descomposición de Tareas (EDT)

La siguiente lustración (Ilustración 4) muestra la estructura de descomposición de trabajo de este proyecto. Está dividido en cuatro Sprints, como se extrae de la metodología anteriormente mencionada, y se ha reservado un bloque para las tareas transversales. En cada Sprint asignamos también los requisitos correspondientes basados en su prioridad y dependencias.

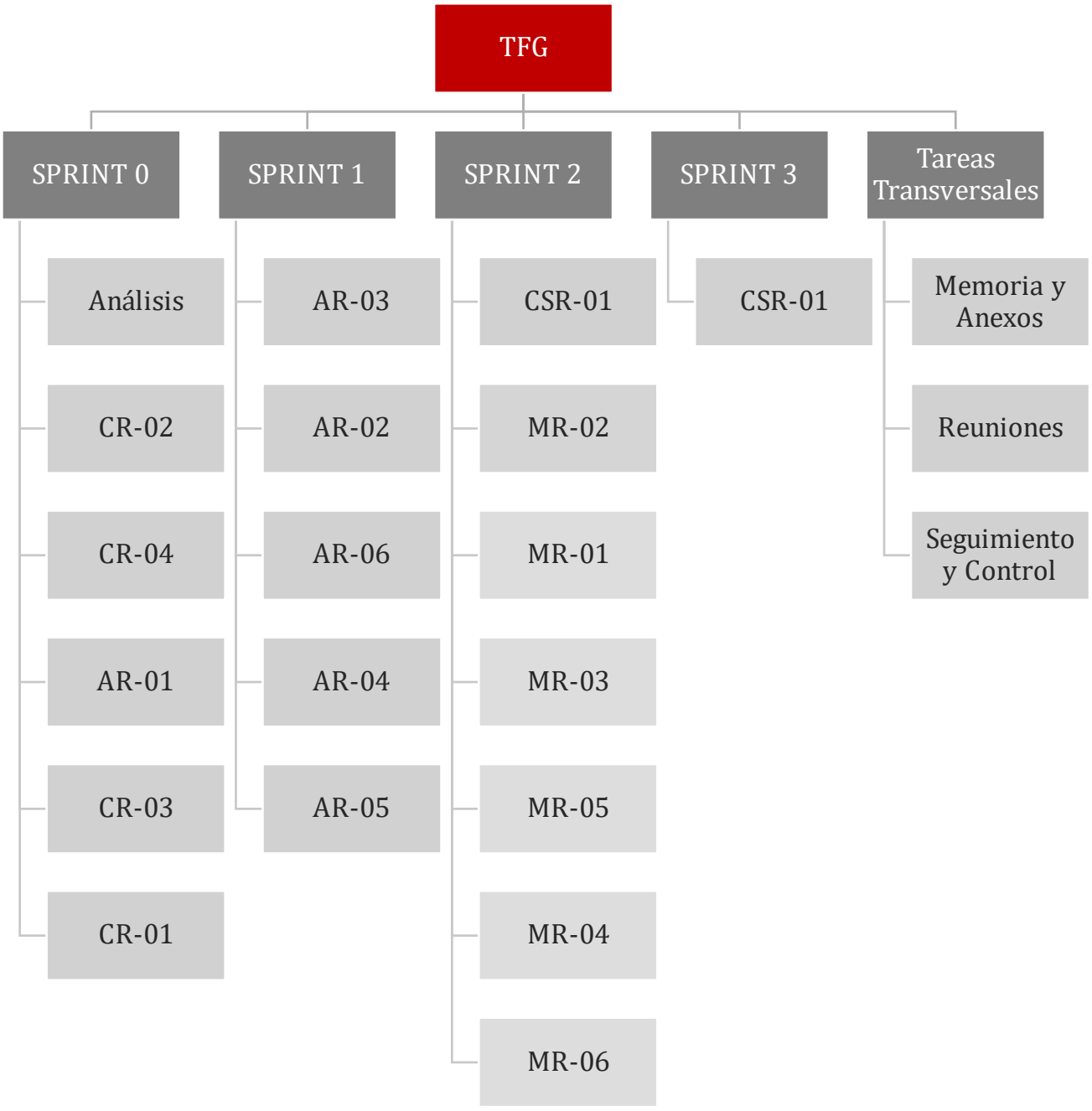


Ilustración 4 - Descomposición de Tareas (EDT)

Calendario y Horario de Trabajo

El presente Trabajo de Fin de Grado se comenzó durante el periodo de prácticas extracurriculares en la empresa el 4 de noviembre de 2019, aunque el convenio de TFG se firmó el 19 de diciembre de 2019.

Esto es debido a que el convenio de TFG no se podía firmar hasta que la Comisión Académica aprobara la asignación de los TFG. En cualquier caso, la fecha en la que se comenzó a trabajar en el presente Trabajo de Fin de Grado se corresponde con el 4 de noviembre.

La otra limitación que nos impusimos fue defender el TFG en la convocatoria de febrero, cuyas fechas de depósito eran del 17 al 19 de febrero.

Noviembre						
Do.	Lu.	Ma.	Mi.	Ju.	Vi.	Sá.
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Diciembre						
Do.	Lu.	Ma.	Mi.	Ju.	Vi.	Sá.
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Enero						
Do.	Lu.	Ma.	Mi.	Ju.	Vi.	Sá.
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

	Lu.	Ma.	Mi.	Ju.	Vi.
8:00 - 9:00					
9:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00					

Ilustración 5 -Calendario y Horario de Trabajo

Diagrama Gantt

La siguiente tabla (Ver Ilustración 6) muestra el diagrama de Gantt del proyecto. En el encontramos la estimación de horas necesarias para completar cada una de las tareas y su distribución en el tiempo. Debido a mi inexperiencia realizando planificaciones la estimación horaria se hizo en base a la complejidad que en un principio se presuponía para cada requisito.

Diagrama Gantt													
Paquetes de trabajo		Noviembre				Diciembre				Enero			
		S1 25h	S2 25h	S3 25h	S4 25h	S1 15h	S2 20h	S3 25h	S4 0h	S1 15h	S2 25h	S3 25h	S4 25h
Sprint 0	Análisis	20h											
	CR-02	5h											
	CR-04		3h										
	AR-01		2h										
	CR-03		5h										
	CR-01		5h										
Sprint 1	AR-03		10h										
	AR-02			20h									
	AR-06			5h	15h								
	AR-04				10h	10h							
	AR-05					5h	15h						
Sprint 2	MR-02							15h					
	MR-01							10h					
	MR-03									15h	15h		
	MR-05										10h	15h	
	MR-04											5h	
	MR-06											5h	15h
Sprint 3	CSR-01						5h						10h
Tareas Transversales	Memoria y Anexos												
	Reuniones												
	Seguimiento y Control												

Ilustración 6 - Diagrama Gantt

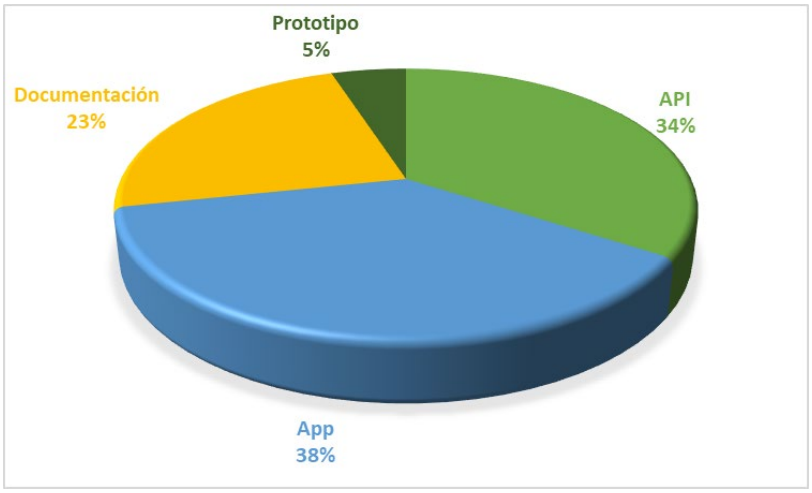
Tiempo Estimado Semana no laboral

Estimación temporal de tareas

En la siguiente tabla (Ver Tabla 8) podemos ver la estimación temporal de todas las tareas del proyecto y seguidamente un gráfico de distribución de peso de cada una de las partes del proyecto. (Ver Ilustración 7)

PLANIFICACIÓN DE TIEMPOS		
Tarea		**Horas Estimadas**
Sprint 0	Análisis	20 horas
	CR-02	5 horas
	CR-04	3 horas
	AR-01	2 horas
	CR-03	5 horas
	CR-01	5 horas
	Total:	**40 horas**
Sprint 1	AR-03	10 horas
	AR-02	20 horas
	AR-06	20 horas
	AR-04	20 horas
	AR-05	20 horas
	Total:	**90 horas**
Sprint 2	MR-02	15 horas
	MR-01	10 horas
	MR-03	30 horas
	MR-05	25 horas
	MR-04	5 horas
	MR-06	20 horas
	Total:	**105horas**
Sprint 3	CSR-01	15 horas
	Total:	**15 horas**
Tareas Transversales	Memoria y Anexos	30 horas
	Reuniones	5 horas
	Seguimiento y Control	15 horas
	Total:	**50 horas**
	TOTAL:	**300 horas**

Tabla 8 - Tabla de Tiempos



Metodología de control de versiones

En este punto explicaré qué servicios he utilizado para realizar un control de versiones, parte de un requisito impuesto en la fase de análisis por la empresa.

En primera instancia he creado dos repositorios de "GitHub" para el tener un control de versiones del código. Estos repositorios a su vez se usarán internamente en la empresa para que cualquier futuro desarrollador tenga el código tanto de la aplicación móvil como el de la aplicación web.

En cuanto a la documentación también se han creado dos repositorios, uno en "Google Drive" y otro en "Microsoft Sites". El primero se ha usado como repositorio personal para tener siempre una copia de seguridad de todo lo que iba haciendo del proyecto. El segundo se ha utilizado para comunicación con el tutor académico.

Plan de contingencia

En este punto se analizarán en la primera tabla (Ver Tabla 9) aquellos riesgos que se han identificado previamente al desarrollo del proyecto junto con su impacto estimado. En la segunda tabla (Ver Tabla 10) se puede ver para cada riesgo determinado cual será la manera de tratarlo.

Identificación de riesgos

Fuente	Riesgo	Impacto
Metodología	Durante el desarrollo pueden aparecer requisitos no contemplados	Medio
Planificación	Superar horas permitidas de TFG.	Medio

Tabla 9 - Tabla de Riesgos

Identificación de soluciones

Fuente	Riesgo	Contingencia
Metodología	Durante el desarrollo pueden aparecer requisitos no contemplados	A cada nuevo requisito se le asignará una prioridad, haciendo primero los requisitos más prioritarios y posteriormente los menos prioritarios.
Planificación	Superar horas permitidas de TFG.	Dejar sin realizar algún requisito bien de los iniciales o bien de los nuevos.

Tabla 10 - Tabla de soluciones riesgos

Tecnologías empleadas

Las tecnologías empleadas en este Trabajo de Fin de Grado vienen heredadas del desarrollo anterior del que se parte en este proyecto, pero además se han añadido más tecnologías complementarias para la planificación y control de versiones. Siguiendo con la división natural del proyecto vamos a determinar primero las tecnologías empleadas en la parte web y en la parte móvil.

Para la parte web se ha utilizado como visor y plataforma web **WordPress**. El uso de este software ha sido de obligada utilización por parte de la empresa ya que lo utilizan usualmente, por eso ha sido, en consecuencia, bien definido como requisito no funcional del proyecto. En cuanto a la API se ha realizado como un "plugin" de **WordPress** y se ha utilizado el lenguaje de programación "**PHP**" para realizarlo ya que es el lenguaje que demanda WordPress.

Puesto que el desarrollo se ha realizado en local se ha tenido que levantar un servidor. Para ello he utilizado el programa "**Xampp**" ya que es bastante intuitivo y sencillo de usar. La elección de este software viene de nuevo impuesto por la empresa. Este programa levanta un servidor "**Apache**" y una base de datos "**MySQL**".

Para el testeo de las peticiones se utilizará el software "**Postman**" que permite realizar peticiones "get" y "post" a una "url" pudiendo editar sus valores en la petición.

Para la parte móvil se ha utilizado **Visual Studio** para programar con **Xamarin**. Se ha elegido este lenguaje ya que, como en los casos anteriores, se ha heredado de la aplicación origen. Esta aplicación dispone de base de datos, y se eligió usar una base de datos "**SQLite**".

Para la gestión del proyecto se ha utilizado además "**Trello**" un gestor de tablones Kanban. Se ha utilizado "**Trello**" ya que es el mismo programa de gestión que se utilizaba en la empresa.

Para la comunicación con el tutor de empresa, debido a que trabaja desde Madrid, se ha utilizado "**Skype**" tanto para mandarnos mensajes como para realizar las reuniones.

Capítulo 2

Este capítulo se centrará en la configuración de todos los entornos de desarrollo, así como todos los requerimientos cuya prioridad sea igual a 0.

Sprint 0

Identificación

Como se ha comentado al inicio del capítulo los requisitos que se han identificado para ser realizados en este sprint son los siguientes: **Análisis, CR-02, CR-04, AR-01, CR-03, CR-01.**

Este primer "sprint" tiene como objetivo poner a punto todos los entornos con los que se va a trabajar, así como sus proyectos, y hacer un revisionado del código con el que posteriormente se irá trabajando.

Proceso

Tarea 0 – Planificación y definición de entregables principales. (Análisis).

La primera tarea de este sprint implica todo el proceso de análisis del proyecto, dividido en varias partes.

Definición de Antecedentes, Actores involucrados, Punto de Partida, Definición de sistema (Engloba la Captura de Requisitos, el Alcance, Definición de Entregables, Exclusiones y Dependencias) y Planificación (Engloba el Esquema de Descomposición de Tareas, Estimación Temporal, Calendario y Horario de Trabajo, Diagrama Gantt, Definición de Metodologías, Plan de contingencia y Tecnologías empleadas).

Tarea 1 – Creación infraestructura pruebas (CR-02)

En esta primera tarea nos centraremos en el requisito con código **CR-02**, crear una infraestructura de pruebas.

Hasta ahora habíamos estado trabajando sobre la misma solución en "Visual Studio", ya que era la solución que estábamos utilizando para realizar la aplicación móvil anteriormente mencionada para el Ayuntamiento de Ezcaray, y sobre la misma página web en WordPress, realizada, de la misma manera, para el Ayuntamiento de Ezcaray. Por lo que nuestro primer paso era crear una infraestructura de pruebas, para evitar alterar el producto del anterior proyecto, pudiendo hacer pruebas durante los "sprints" y crear un prototipo final.

Ante esto lo primero que hicimos fue pensar qué era realmente necesario y qué no. En este punto fuimos conscientes de que no disponíamos en la empresa de ninguna forma de exportar tanto el código de la aplicación móvil como la configuración, datos, los plugin y el entorno de la página web y de la API.

Por lo tanto, separamos esta tarea en dos tareas independientes, la configuración del entorno de la aplicación móvil, y homólogamente respecto a la página web y API. Además, se añadieron dos **requisitos no funcionales,** requisitos **NFR-04 y NFR-05 (Ver Tabla 9)** al proyecto, realizar un documento para la explicación de cada entorno, de ese modo, se crearán dos nuevos entregables al proyecto, completando dichos documentos durante la realización de los "sprints".

Tarea 2 – Crear entorno aplicación (CR-02 A)

Debido a que la naturaleza del proyecto abarca tanto una aplicación móvil como una página web decidimos centrarnos primero en la aplicación móvil, ya que era el entorno con el que más familiarizado estaba, al haber estado trabajando con él en el periodo de prácticas.

Creamos una nueva carpeta en el disco y una nueva solución en "Visual Studio". Duplicamos el proyecto en otra solución y aquí encontramos un problema, el código estaba lleno de referencias que necesitábamos cambiar. Tras solucionar este problema la aplicación finalmente funcionó. Al duplicarla nos dimos cuenta de que había una serie de recursos que debían ser únicos para cada aplicación:

- **Nombre de la aplicación e icono:** Nombre e icono identificadores de la aplicación móvil que se mostrarán en el menú del dispositivo móvil y en sus detalles.
- **Gif de pantalla de carga:** gif que se muestra en la pantalla de carga que identifica a la propia aplicación móvil.
- **Icono superior e inferior del menú lateral de la aplicación móvil:** iconos de personalización del menú de la aplicación.
- **Imagen de fondo del menú lateral:** imagen de personalización del menú.

Estos datos debían de tenerse en cuenta a la hora de realizarse el requisito **CSR-01**, creación del prototipo final.

Asimismo, se creó un repositorio en GitHub y se vinculó con la solución para tener un control de versiones, como se ha explicado en la sección del presente documento Metodología de control de versiones.

Tarea 3 – Infraestructura web (CR-02 B) y Creación entorno WordPress (CR-04)

Una vez acabada la anterior tarea, pasamos a centrarnos en configurar el entorno de la página web y la API.

Para ello instalamos una serie de herramientas necesarias para su ejecución mencionadas en la sección de Tecnologías Empleadas y creamos una nueva base de datos y una nueva página web y en este punto decidimos realizar el requisito **CR-04**, clonar entorno WordPress, ya que crear el entorno de WordPress junto con su base de datos era parte indispensable en la creación de la infraestructura web.

Por lo que procedemos a clonar a mano todos los **datos**, **metadatos** y **attachments** de la página web realizada en el periodo de prácticas.

En este punto somos conscientes de que configurar una página web de cero es una tarea bastante ardua y por lo tanto decidimos que sería necesario investigar una manera de exportar todos estos datos de una manera más sencilla. Así que conseguimos extraer un nuevo **requisito,** requisito **AR-07 (Ver Tabla 9)** de este punto.

Tarea 4 – Clonación de plugins (AR-01)

Para el siguiente paso decidimos realizar el requisito **AR-01**, Estandarización de Plugins. En este punto se analizó qué "plugins" de los que estaban instalados en la página web eran realmente necesarios y se eligieron los que están referenciados en el documento de Configuración de WordPress **(Ver Anexos).** Tras esto se instalaron y se configuraron, plasmando todo ese proceso en el documento anteriormente mencionado.

Tarea 5 – Conectar api-app. (CR-03)

Tras levantar el servidor en local y tras configurar la base de datos de WordPress, decidí que era el momento para conectar la API con la aplicación móvil. El proceso era sencillo pero tedioso, ya que, para una simple conexión, tal y como lo teníamos planteado, había que cambiar varias referencias a la "url" y al token de autenticación en la aplicación móvil.

Lo primero que se hizo fue generar un nuevo usuario en WordPress con rol de administrador para que fuera el quien se comunicara con la base de datos. Después generamos el token de autenticación, con los datos (nombre de usuario y contraseña) del usuario que habíamos creado. Teniendo ya el token de autenticación y la "url", pasamos a introducirlas en la app.

Tras cambiar todas las referencias a ambas configuraciones la conexión no funcionaba, por lo que se perdió un día de trabajo en solucionarlo, pese a ello, no supuso una desviación de la planificación ya que para esta tarea se había planificado tardar un día. Lo que pasaba es que, al trabajar en localhost, al levantar un emulador Android desde "Visual Studio", el emulador está en otra red y hay que utilizar una "ip" específica para que se comuniquen. Tras arreglar esto la conexión finalmente funcionó. Todos estos pasos se añadieron al documento de Configuración Aplicación Móvil. **(Ver Anexos)**

Tarea 6 – Limpieza de código (CR-01)

Una vez realizada la conexión se decidió, para facilitar el trabajo, eliminar todo código que no se use o bien que esté comentado. Se realizó en este momento y no antes ya que preferíamos eliminar código en un entorno seguro, sin comprometer la integridad del proyecto anterior al pertenecer a un cliente real, el Ayuntamiento de Ezcaray.

Teníamos muchos métodos, clases e incluso ficheros que se habían quedado obsoletos, debido a que, cuándo se empezó el proyecto, la inexperiencia tanto de mi compañero al realizar el plugin como mía al realizar la aplicación móvil, hizo que las dos soluciones se llenaran de contenido que no era necesario, o que se creó a manera de prueba.

Además, como ya mencioné anteriormente en antecedentes del presente documento, a que a pesar de que era el campo en el que había estado trabajando en el periodo de prácticas, debido al plazo de entrega tuvo que intervenir mi superior en el proyecto, por lo que había muchos ficheros que se crearon pensando que eran necesarios pero al final no servían para nada. A esto se le suma además el ruido que "Xamarin" genera solo.

Se tardó un día más de lo planeado inicialmente en esta tarea debido a que no fuimos realmente conscientes de todo aquello que había que eliminar hasta que no estuvimos realmente frente al problema, ya que no solo se eliminó código, sino que también hubo que eliminar imágenes, clases inútiles, y fuentes.

Review

Una vez acabado este "sprint" se determinarán los requisitos que se han quedado sin hacer y el porqué de estos, así como los nuevos requisitos que se han extraído de este "sprint". (Ver Tabla 11)

En este "sprint", se consiguieron realizar todos los requisitos que nos propusimos, en parte, porque este primer "sprint" correspondía con la configuración inicial del proyecto.

Aun así, pese a ser una primera iteración del proyecto nos hemos encontrado tres nuevos requisitos, dos no funcionales y uno funcional, que siguiendo la nomenclatura son los siguientes:

Código	Titulo	Prioridad
NFR-04	Documentación de creación infraestructura web.	-
NFR-05	Documentación de creación de entorno de aplicación móvil.	-
AR-07	Exportar arquitectura modelo de datos aplicación web.	1

Tabla 11 - Requisitos resultantes Sprint 0

NFR-04 - Documentación de creación infraestructura web.

Requisito detectado en este sprint, útil para facilitar la creación y configuración de una infraestructura web en futuros proyectos.

NFR-05 - Documentación de creación de entorno de aplicación móvil.

Requisito detectado en este sprint, útil para facilitar la creación y configuración del entorno de la aplicación móvil y su despliegue en futuros proyectos. (Ver Anexos)

AR-07 - Exportar arquitectura modelo de datos aplicación web.

Se buscará la manera de exportar el modelo de datos del gestor de contenidos, es decir, exportar los tipos de datos de WordPress. (Ver Anexos)

Aunque se hará una revisión más exhaustiva en el Capítulo 6 sobre la relación entre tiempo estimado y tiempo total dedicado a realizar cada Sprint, destacar que la realización de este Sprint en concreto nos ha llevado un día más de lo planeado. Este hecho unido con la aparición de tres nuevos requisitos nos va a hacer muy difícil ajustarnos a la planificación que habíamos planteado en un principio.

Capítulo 3

Este capítulo se centrará en todos los requerimientos cuya prioridad sea igual a 1 y los requisitos cuya prioridad sea igual a 0 que no hayan podido realizarse en el sprint anterior.

Sprint 1

Identificación

Como se ha comentado al inicio del capítulo los requerimientos que se han identificado para ser realizados en este sprint son los siguientes: **AR-03, AR-02, AR-06, AR-04 y AR-07** ya en el anterior sprint no quedó ningún requisito con prioridad igual a 0 sin hacerse. El último requisito viene derivado del sprint anterior, ya que al otorgarle prioridad 1 será realizado en este sprint.

Proceso

Tarea 1 - Definir estándar para la estructura de datos de la API. (AR-03)

Esta tarea tendrá como objetivo definir la estructura de como los datos se envían a la aplicación móvil desde el entorno web. Hasta ahora se enviaban todos los datos en formato "json" dentro del cuerpo de una petición post que realizábamos a una "url" habilitada para devolver esos datos. (Ver Ilustración 8)

```
http://(url_host)/(nombre_sitio_web_WordPress)/wp-json/jwt-auth/v1/data
```

Ilustración 8 - Petición original

La cantidad de información que se enviaba en formato "string" era muy grande tardando mucho tiempo tanto en generar los datos en la aplicación web, como a la hora de leerlos por parte de la aplicación móvil. Ante esto lo que hicimos fue dividir esa información en dos bloques.

Un bloque para el "media" (attachments), archivos como "pdfs", imágenes, etc. Y otro bloque para los datos. El primer bloque contendría toda la información relativa a todos los media del gestor de contenido. Para cada uno de los archivos almacenados en media extraemos de WordPress y enviamos por medio de la API:

- **Id:** "Id" determinado por WordPress que identifica a ese archivo dentro de la base de datos.
- **URL:** "Url" del archivo media dentro de WordPress.
- **Mime_type:** Indica cual es el tipo y extensión del archivo.
- **Tamaño**: Indica el tamaño que pesa en MB el archivo.
- **Base64**: Convertimos el fichero a base64 y lo almacenamos en este campo.

Habilitamos una "url" para poder realizar otra petición post y así pedir los datos en dos partes. Generando de este modo dos "url" de petición en vez de una, estas "url" eran las siguientes: (Ver Ilustración 9)

```
http://(url_host)/(nombre_sitio_web_WordPress)/wp-json/jwt-auth/v1/data
http://(url_host)/(nombre_sitio_web_WordPress)/wp-json/jwt-auth/v1/attachments
```

Ilustración 9 - Urls nuevas

Tras probar esto nos dimos cuenta de que, en vez de ser beneficioso en cuanto a tiempo, este cambio ha causado todo lo contrario, ya que tardaba incluso más. No nos detuvimos en determinar el origen del problema, sino que directamente se buscó una solución. (Aunque la intuición parece indicar que estábamos descargando un volumen similar de datos con un número mayor de peticiones)

El primer paso fue dividir el data del metadata, de la misma manera que se había separado los attachments del total, ya que se pensó que incluir en el mismo "json" los datos de contenido junto con los datos de configuración de la propia aplicación móvil, (explicados en Punto de Partida), no era del todo buena idea, ya que los datos de configuración debían tener una prioridad mayor a la hora de ser cargados en la aplicación que los datos propios de contenido.

Por lo tanto, se generó otras dos "url" diferentes para devolver en una petición post los "json" de estas dos partes. Teniendo finalmente tres "url" de petición diferentes, una para **attachments**, otra para **data** y otra para **metadata**.

```
http://(url_host)/(nombre_sitio_web_WordPress)/wp-json/jwt-auth/v1/data
http://(url_host)/(nombre_sitio_web_WordPress)/wp-json/jwt-auth/v1/metadata
http://(url_host)/(nombre_sitio_web_WordPress)/wp-json/jwt-auth/v1/attachments
```

Ilustración 10 - Url de petición finales

Este cambio dio lugar a un **requisito,** requisito **MR-07 (Ver Tabla 10)** que no teníamos contemplado que es el crear un servicio en la aplicación móvil para leer los datos de la petición, ya que en vez de tener una sola "url", ahora teníamos tres diferentes.

Aun así, pese a dividir la petición en tres peticiones independientes seguíamos con el mismo problema el tiempo de carga no se reducía, por lo que, gracias a una idea de mi tutor en la empresa, se decidió que en vez de generar en cada petición los datos a enviar, se generaría tres archivos "json" con los sus datos.

Hasta ahora en cada petición generábamos los datos que se solicitaban por cada petición (data, metadata y attachments) enviándolos en formato "json" a la aplicación móvil. Ahora se crearon tres nuevas "url" que generaban los datos de las tres peticiones, creando un archivo por cada petición y almacenando los datos dentro de cada archivo. Asimismo, las peticiones de solicitud (Ilustración 10) en vez de generar los datos y enviarlos lo que hacen ahora es leer directamente el fichero y devolver su contenido.

Se determinó que la aplicación únicamente conocería las "url" de petición, no de generación, delegando la responsabilidad de actualizar los archivos a la persona que gestione la infraestructura web, mediante el programa "Postman", que permite realizar peticiones "get" y "post" a una "url" pudiendo editar los valores de la petición.

Esto mejoró significativamente los tiempos de espera, ya que nos evitábamos el tiempo de generación del "json", pero causó que tardáramos un día más de lo esperado al no tenerlo contemplado.

Tarea 2 - Mejora del rendimiento de carga de la API (AR-02)

Así pues, se generaron tres "url" para devolver todos los datos y tres "url" para generarlos, pero aun así seguíamos teniendo un problema con el archivo attachments. Tanto la generación como la petición seguía tardando mucho debido a que los archivos más grandes estaban lastrando el tiempo.

La aplicación contiene múltiples archivos para diferentes propósitos, por ejemplo, tenemos imágenes para personalizar contenidos, archivos "pdf" para extender la información mostrada, etc.

El motivo del lastre era el campo del "json" en el que tenemos el archivo convertido a base64. El lastre era tanto al generarlos, debido al alto coste de convertir los ficheros a base64, como al solicitarlos, debido al alto coste de enviar todos los archivos, independientemente de su tamaño.

La solución que tomé fue poner una cota máxima de tamaño de archivo haciendo que los que superaran ese tamaño no se les rellenaría el campo base64. Se hicieron diferentes pruebas con diferentes cotas y finalmente decidimos establecer la cota en 250MB ya que seguir aumentándola no mejoraba realmente el rendimiento.

Se definió asimismo que si el archivo en la petición no disponía del campo en base64 relleno se descargaría posteriormente si se solicitase desde la aplicación. El archivo se descargaría desde su "url" propia, esto dio lugar a un nuevo **requisito,** requisito **MR-08 (Ver Tabla 10)** que compete a la parte de la aplicación móvil, que consistirá en hacer una implantación de servicio de descarga de archivos.

Tarea 3 - Estandarización de pantallas de visualización/tipo. (AR-06)

Como ya se ha comentado en la descripción del requisito a acometer en esta tarea, el objetivo es poder definir desde el CMS qué vista, del catálogo de vistas que dispone la aplicación, quiere el usuario que se utilice para cada tipo de dato.

En la aplicación móvil tenemos dos tipos de transiciones entre las vistas: abrir una lista de elementos, abrir un elemento en detalle. Adicionalmente tenemos dos transiciones especiales: abrir un contenido web y abrir la visualización de tiempo. En este punto vamos a tratar las dos primeras, que serán el objeto de parametrización de esta tarea.

La vista de lista se utiliza para mostrar varios elementos a la vez, elementos que serán mostrados con la misma plantilla dentro de la lista. Dentro de las listas podemos encontrar varios tipos de lista diferentes como las mostradas a continuación.

Lista estándar **Mapa** **Calendario** **Carrusel**

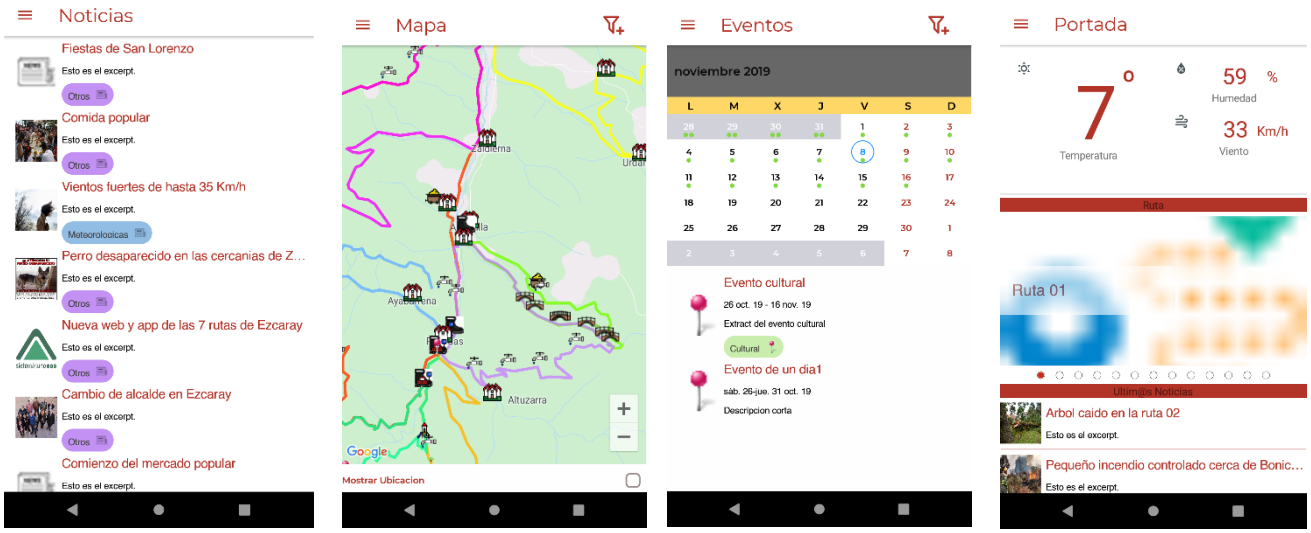


*Ilustración 11 - Vistas de listas**

La forma de representar cada elemento dentro de la lista **(Ver Ilustración superior, primera y tercera imagen)** son lo que referimos como **"Ventana Template Lista". "Ventana detalle"** es la vista individual de un elemento de la lista, esta visualización puede ser única o genérica.

Las listas genéricas pueden ser únicas para cada tipo, ya que, con cambiar únicamente la "**Ventana Template Lista"** se generan listas completamente diferentes. Estas listas serán las que elegiremos mayoritariamente para mostrar los datos. La vista en detalle refiere a la vista en concreto de un elemento.

Una vez teniendo claro que tipos de vistas tenemos disponibles en la aplicación se comenzó la realización de este requisito.

Lo primero que se hizo fue definir para cada tipo de dato (metadata) cuál de las vistas en lista que tenemos creadas en "Xamarin" se le aplicará, que tipo de "template" se le aplicará a cada uno de los elementos de esa lista y que ventana de detalle se le aplicará a la misma, como se ha explicado anteriormente. El objetivo final es hacer que la aplicación se comporte de la manera que se comporta WordPress por defecto, es decir, que cada tipo de dato (metadata) tenga asignada una visualización concreta de sus elementos en conjunto y de sus elementos individuales.

*Nota: Las ilustraciones anteriores son del proyecto para el Ayuntamiento de Ezcaray

Por lo que se añadió a cada tipo de dato, los tres nuevos campos que corresponden a los tres tipos de visualizaciones previamente mencionados.

Ilustración 12 – Campos estandarización visualización tipo

Estos datos se enviarán mediante la petición a la "url" metadata y la aplicación móvil deberá ser capaz de gestionarlos y actuar en consecuencia.

Adicionalmente se añadió otro campo para definir desde la aplicación web para cada tipo de dato cuál será su tipo de comportamiento. Hemos llamado tipos de comportamiento a aquel comportamiento especial que puede tener un tipo de dato en concreto. Tipos de comportamiento puede haber tantos como se definan, por lo que lo primero que había que hacer era definir qué comportamiento especial podíamos encontrar para cada tipo en nuestro proyecto y encontramos los siguientes comportamientos especiales.

- **visibleMapa**: que tipos de datos se mostrarán en la vista de lista en formato mapa, si es que hay alguno.

- **muestraFiltrosDrawer**: que tipos de datos tendrán un menú lateral para poder filtrar los elementos de esa lista. El filtro era una funcionalidad que estaba ya realizada y que nos permite filtrar por fecha, por etiquetas, y por tipo, es decir, que solo se mostrarán los datos en la vista que determinemos en el filtro. Por defecto se muestran todos.

- **renderMapaPolyline**: sirve para mostrar rutas en el mapa, que pese a que no se usará en este prototipo era necesario añadirlo para futuros proyectos.

- **visibleEnCalendario**: que tipos de datos se mostrarán en el calendario de la aplicación.

- **visiblePortadaCarrusel**: que tipos de datos se mostrarán en formato de lista carrusel en la portada de la aplicación.

- **visiblePortadaProximos**: que tipos de datos se mostrarán en formato de lista genérica en la portada de la aplicación en la sección llamada "Proximxs" + nombre del tipo. (Ej. Próximos Eventos).

- **visiblePortadaUltimos**: que tipos de datos se mostrarán en formato de lista genérica en la portada de la aplicación en la sección llamada "Últimxs" + nombre de tipo. (Ej. Últimas noticias).

Para este propósito hemos optado por poner un campo que contendrá "flags" (**1** ó **0**) según si tiene un comportamiento especial concreto o no. Se han reservado 32 bits para determinar su comportamiento, aunque por el momento solo hemos definido y usado siete bits que corresponden con los comportamientos especiales ya explicados. Esta es la asignación de los comportamientos con sus bits:

- visibleMapa = 1, (Bit 0)
- muestraFiltrosDrawer = 32, (Bit 5)
- renderMapaPolyline = 64, (Bit 6)
- visibleEnCalendario = 2048, (Bit 11)
- visiblePortadaCarrusel = 4096, (Bit 12)
- visiblePortadaProximos = 8192, (Bit 13)
- visiblePortadaUltimos = 16384 (Bit 14)

Como se puede ver, se han dejado bits sin asignar para poder definir en un futuro nuevos comportamientos especiales.

Tarea 4 – Creación de menús dinámicos y configuración total (AR-04)

A petición de la empresa como cliente se pidió que los menús de la aplicación (menú superior y lateral) fuesen dinámicos, pudiendo añadir o eliminar contenidos de una manera sencilla sin tener que generar una "apk[4]" de actualización de la aplicación

Frente a este requerimiento se decidió que la única manera de hacerlo era que los menús se determinaran desde la API. Creamos un nuevo tipo de dato en WordPress "Configuración" que solo tendría un título y un "content". El "content" contendrá la configuración de cada punto en el menú en formato "json".

Se definió una estructura "json" en la cual cada elemento dentro de un menú estaría formado por los siguientes campos:

- **Tipo**: será un numero entre 1 y 5, ambos incluidos, cada número indica una opción a realizar:

  1- **abrirTipo**: esta opción abre un tipo en concreto de datos, esto quiere decir que muestre todos los datos relativos a un tipo de dato en formato lista genérica.

  2- **abrirTiposFlags: e**sta opción comprueba que bit de los "flags" (anteriormente mencionados) está activo para todos los tipos de datos. Se extraerán todos los datos de la base de datos móvil de los tipos que tengan ese "flag" activo, mostrándolos todos en una vista. Esto se ha hecho con el objetivo futuro de integrar varios tipos de datos dentro de una misma vista.

[4]APK: Un archivo con extensión .apk es un paquete para el sistema operativo Android

3- **abrirIDElemento**: esta opción abre la vista en detalle de un dato en concreto. Por defecto se le aplicará la vista en detalle de su tipo asociado.

4- **abrirPagView**: esta opción abre una vista en concreto sin ninguna relación con ningún tipo como es el caso del componente que utilizamos de meteorología, que no está asociado a ningún tipo de dato, sino que obtiene la información de la API oficial de AEMET.

5- **separador:** esta opción sirve para mostrar una opción de menú que no tiene ninguna función más que delimitar secciones dentro del propio menú.

- **Título**: nombre que se mostrará en el menú.
- **idIcono**: "Id" del attachment de la imagen que se mostrará como icono en el menú.
- **Argumentos**: se seleccionará en función de su tipo uno de los argumentos siguientes:
    - **modifLong:**
        - Si es **Tipo 1** refiere al tipo de dato a abrir, es decir que se abrirán en formato de lista todos los datos de un tipo de dato en una vista.
        - Si es **Tipo 2** refiriere a que bit de los "flags" (anteriormente mencionados) se comprobará que este activo para todos los tipos de datos.
        - Si es **Tipo 3** indica el "id" individual con el que se referencia en la base de datos de WordPress del elemento a abrir. Este "id" coincidirá con el mismo "id" en la base de datos local de la aplicación móvil.
    - **modifString:**
        - Si es **Tipo 2** será la vista a aplicar, esto se hará por si queremos aplicar a un tipo otra vista diferente de la que tiene asociado.
        - Si es **Tipo 3** abre aquel dato cuyo campo "url" coincida con la "url" introducida en este parámetro. Se ha integrado esta opción por si por un casual fallara la búsqueda por "id" o bien no se pueda extraer el "id" de un dato en concreto de la base de datos de WordPress.
        - Si es **Tipo 4** ruta interna completa de la vista a abrir. Esto se ha hecho por si queremos mostrar una vista en concreto, como por ejemplo la vista en detalle de la meteorología que no está ligada a ningún tipo de dato.

Adicionalmente, si el elemento es de **Tipo 2**, se le podrá añadir los siguientes argumentos dentro de la definición de un elemento del menú:

- **modifBool:** nos indica si se muestra filtro en la vista o no. Solo puede tomar dos valores, "true" o "false".
- **modifString2:** indicará la vista en detalle de un elemento individual a aplicar. Esto se hará por si queremos aplicar a un dato otra vista diferente de la que tiene asociado su tipo de datos. Si este campo no existe se aplicará la vista asociada a su tipo de datos.

Aquí podemos ver un ejemplo:

```
{'tipo':1,'titulo':'Eventos','modifString':'eventos','idIcono':222}
```

Ilustración 13 - Ejemplo de objeto de menú

Este nuevo tipo se comportará como un objeto de data normal, es decir, que se solicitará mediante la petición a la url de "data".

El siguiente paso en la configuración de menús se hará en el siguiente Sprint, que corresponde a la tarea asociada con el requisito **MR-05**.

Tarea 5 – Exportar arquitectura modelo de datos aplicación web (AR-07)

Se buscará, como se indicó en la definición de este requisito, la manera de exportar el modelo de datos del gestor de contenidos, es decir, exportar los tipos de datos de WordPress, con el propósito de poder ser reutilizados en futuros desarrollos.

Para este propósito se utilizaron las funciones propias de los "plugins" utilizados ya que al analizarlos nos percatamos de que permitían una exportación bastante natural. Lo que se hizo fue hacer pruebas con la base de datos de la aplicación web realizada para el Ayuntamiento de Ezcaray al considerarse que estas pruebas no comprometerían la integridad de ese trabajo. Se determinó si era suficiente o no la exportación que nos brindaba estos "plugins".

Los "plugins" usados son **Custom Post Type UI** y **Advanced Custom Fields**:

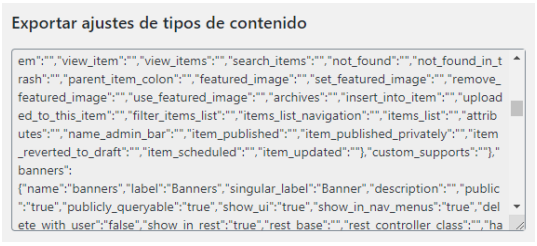


Ilustración 14 - Exportación de tipos

❍ El primero exporta todos los tipos de datos de la aplicación web, pero no su contenido. Esto era un acierto ya que nos servirá para exportar elementos comunes ya configurados como por ejemplo Noticias, Eventos, etc.

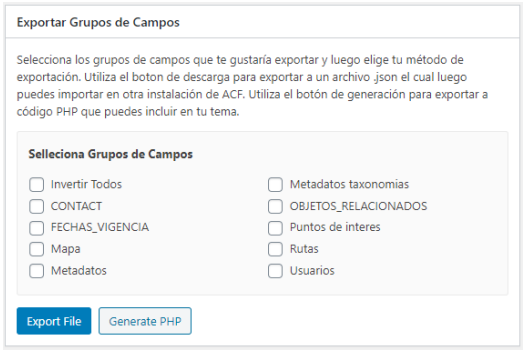


Ilustración 15 - Exportación total de tipos

❍El segundo exporta los campos personalizados que se utilizan en cada tipo de dato. Esto era muy útil ya que extendía la funcionalidad del anterior, logrando extraer todo lo relativo a los tipos de datos de una manera sencilla.

Review

Una vez acabado este "Sprint" analizaremos los requisitos que se han quedado sin hacer y el porqué de estos, así como los nuevos requisitos que se han extraído de este "Sprint". (Ver Tabla 12)

En este sprint, se realizaron los requisitos: **AR-03, AR-02, AR-04, AR-05, AR-06** y **AR-07,** siendo este último requisito extraído en el **Sprint 0**.

En este Sprint finalmente se ha tardado un día menos de lo planeado a pesar de que se ha introducido un requisito que no teníamos contemplado. Esto ha sido en parte gracias a que se decidió dejar sin hacer el requisito **AR-05,** ya que, al aparecer dos nuevos requisitos, y pese a que alguno de los requisitos nos llevó menos tiempo del estimado, no teníamos la certeza de poder ajustarnos a las horas totales marcadas.

Además, no era un requisito tan prioritario para el cliente y eliminarlo nos brindaba 40 horas de margen para dedicar a otros requisitos de nueva aparición, ya que eliminarlo suponía eliminar también el requisito **CR-07** al estar estos dos requisitos estrechamente ligados.

Los dos requisitos identificados en este sprint son los siguientes:

Código	Titulo	Prioridad
MR-07	Unificar métodos y peticiones del servicio de conexión con la API.	2
MR-08	Implantación de servicio de descarga de archivos.	2

Tabla 12- Requisitos resultantes Sprint 1

MR-07 Unificar métodos y peticiones del servicio de conexión con la API:

Este requisito se identificó debido a los cambios que se originaron en la **Tarea 1** de este "sprint"**.** Al hacer cambios en la API debía de verse reflejado ese trabajo en la aplicación móvil, ya que si no la conexión entre ambas partes estaría rota.

MR-08 Implantación de servicio de descarga de archivos:

Al no enviar en la petición post el campo "**base64**" de los archivos hemos de implantar un servicio de descarga independiente de la API, de los ficheros que se soliciten que no tengan base64, mediante la descarga directa desde su "url".

Capítulo 4

Este capítulo se centrará en todos los requerimientos cuya prioridad sea igual a 2 (que corresponden la gran mayoría a requerimientos que competen a la Aplicación móvil) y los requisitos cuya prioridad sea igual a 1 que no hayan podido realizarse en el sprint anterior.

Sprint 2

Identificación

Como se ha comentado al inicio del capítulo los requisitos que se han identificado para ser realizados en este sprint son los siguientes: **MR-02, MR-01, MR-03, MR-05 y MR-04.** En el anterior sprint no quedó ningún requisito con prioridad igual a 1 sin hacerse**.** Además, se realizarán por su prioridad los requisitos **MR-07** y **MR-08** extraídos del Sprint 1.

Proceso

Tarea 1 – Unificar métodos y peticiones del servicio de conexión con la API. (MR-07)

Tras la modificación de la API se tuvo que volver a realizar la conexión con la aplicación móvil ya que las "url" habían cambiado, y, por lo tanto, la aplicación no disponía de datos.

Hasta ahora, la "url" (Ver Ilustración 16) estaba definida como parámetro de entrada fijo en el método de petición de los datos y del mismo modo, el token estaba definido dentro del método, esto hacía que fuera muy difícil el mantenimiento del código.

Tras investigar la mejor opción para implementar esto, se optó por sacar estas cadenas a un archivo independiente y referir posteriormente donde estaban definidas estas cadenas a este archivo. En ese archivo almacené los datos como el token de validación y la cadena de entrada de la "url", que tenía el formato:

```
http://(url_host)/(nombre_sitio_web_WordPress)/wp-json/jwt-auth/v1/data
```

Ilustración 16 - Petición base

Ahora al disponer de tres "url" nuevas (Ver Ilustración 17) lo que se hizo fue almacenar la petición base sin el "data", para poder añadir a esta petición modificada aquello que queramos obtener (**data**, **metadata** o **attachments**).

```
http://(url_host)/(nombre_sitio_web_WordPress)/wp-json/jwt-auth/v1/~~data~~
```

Ilustración 17 - Petición modificada

Tras esto creamos un método de petición independiente para cada petición: (Ver Ilustración 18)

```
diccionarioResult generalRequest(String tipo){
    diccionarioResult result = new diccionarioResult();
    String token = FactoriaGeneral.getDataPersist().getString("TOKEN_WP");
    String url = FactoriaGeneral.getDataPersist().getString("URL_BASE");
    using (var client = new HttpClient()){
        try{
            client.DefaultRequestHeaders.Authorization =
                new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", token);
            HttpResponseMessage response = client.PostAsync(url,
                new StringContent("", Encoding.UTF8, "application/json")).Result;
            Stream dataStream = response.Content.ReadAsStreamAsync().Result;
            StreamReader reader = new StreamReader(dataStream);
            var resultado = reader.ReadToEnd();
            reader.Close();
            result.JSON = resultado;
            return result;
        }
        catch(Exception ex){
            Console.WriteLine("Ha habido un error en la solicitud");
            result.JSON = "";
            return result;

        }
    }
}
```

Ilustración 18 - Método de petición

En este punto tuvimos un problema, se nos había caducado el token que utilizamos para validarnos en la petición ya que cometimos el error de no tener en cuenta (a diferencia de en el caso de la aplicación web del Ayuntamiento de Ezcaray) las acciones a realizar para configurar el plugin de validación y de ese modo hacer que el token no se caducara.

Por lo que se tuvo que generar otro usuario con rol de administrador, modificar el plugin para que no caduque (simplemente cambiar fecha de caducidad a la deseada), volvimos a generar el token con los datos de ese usuario y cambiamos el token en la aplicación móvil.

Tarea 2 – Población BD ejemplo. (CSR-01)

Tras realizar la conexión con la API fuimos conscientes, como ya habíamos determinado en Dependencias, de que no disponíamos de datos suficientes para seguir probando los requisitos siguientes. No podíamos mejorar la conexión de la aplicación con la API si no disponíamos de datos para valorar el rendimiento. De la misma manera no podíamos implantar el servicio de descarga de archivos sin archivos en el CMS.

Al realizar la población de la base de datos encontramos un problema, que en localhost no nos era posible integrar los servicios de mapas de Google para determinar los puntos en el mapa (Latitud y Longitud), por lo que, en primera instancia, decidimos prescindir de esta parte, ya que no disponíamos de mucho tiempo para detenernos en este error, y tampoco teníamos tiempo para localizar o acotar el problema.

Los datos que hemos usado para poblar la base de datos son los que vamos a usar en la realización del prototipo final en el Sprint 3.

Tarea 3– Mejora interacción APP-API (MR-02)

La aplicación base sobre la cual se está trabajando tarda mucho en cargar. Logramos identificar previo al desarrollo de este trabajo que el problema derivaba de que la petición a la API era bastante costosa en tiempo.

Pese a que se había mejorado esta conexión en la Tarea 1 de Sprint 1 reduciéndose notablemente los tiempos de espera, volvimos a estudiar esta conexión identificando dos problemas claramente diferenciados:

- La API no disponía de un control de tiempos y por lo tanto cada vez que se actualizaban los archivos de **data**, **metadata** y **attachments**, se generaba y escribía en ese archivo todos los datos que tuviera la base de datos, a pesar de que no hubieran cambiado.

- La aplicación descargaba todos los datos que tuviera, aunque su base de datos estuviera actualizada respecto a los datos de la API.

Por lo tanto, el objetivo era implantar un control de tiempos en las dos partes. Primero nos centramos en la API, se planteó que, en vez de generarse un archivo con todos los datos, se generarían cuatro archivos. Un archivo que referiría a todos los datos que se han añadido a la API desde hace un año, de manera análoga los otros tres archivos, pero con los datos de hace un mes, una semana y un día.

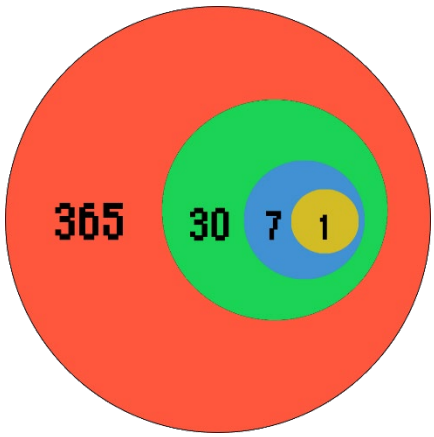


Ilustración 19 - Representación gráfica de estructura de archivos

De esta manera solo se solicitarán los archivos que se necesiten actualizar. Para ello necesitaríamos introducir un dato más en la petición, en concreto decidimos incluir en el "body" el timestamp (Tiempo en segundos) de la última vez que se actualizaron los datos de un archivo en concreto (data, metadata y attachments) en la aplicación. Una vez llega la petición a la API se compara el tiempo enviado con el tiempo actual que sería el tiempo transcurrido entre ambas.

Después pasamos los segundos a días y devolvemos el fichero más próximo que esté por encima de esos días. (Ejemplo: Si hace 8 días que no hemos actualizado los datos devolveríamos el fichero de 30 días, ya que si devolvemos el de los últimos 7 días perdemos la información que se haya introducido el octavo día)

Se adaptó la "url" que generaba estos archivos para que en vez de generar uno generara los cuatro mencionados anteriormente. Para determinar qué datos se añadirán a cada archivo se utilizó herramientas propias de WordPress, como el siguiente comando: (Ver Ilustración 20)

```
'after' => date('F j, Y',$update_since)
```

Ilustración 20 - Comando API

Una vez solucionado el control de tiempos de la aplicación web, solicitando únicamente los datos que no estén actualizados, fuimos conscientes que, si tienes actualizados todos los datos siempre pide por defecto el fichero del ultimo día. Ante esto se pensó añadir en el "header" otro timestamp con la fecha de la última petición a la api desde la aplicación móvil.

De tal manera que, si esa fecha es mayor que la fecha de última modificación del archivo solicitado no enviara el "json" con los datos, sino que enviara la respuesta 304 (*Not Modified*). Esto se hizo enviando en el "header" de la respuesta un timestamp de la última modificación del archivo, guardándolo de esa manera en la aplicación para añadirlo en el "header" de la siguiente petición.

Se modificó el método anterior (***Ilustración 18 - Método de petición***) para incluir en "header" y "body" de las peticiones los "timestamps" de última modificación del archivo solicitado y de última modificación de la base de datos local.

Tarea 4 –Reducción tiempo de carga de la aplicación (MR-01)

Gracias a los cambios realizados en el anterior sprint se logró reducir el tiempo de carga de la aplicación móvil hasta en un 60% respecto a la anterior versión, logrando bajar de un minuto hasta los 25 segundos de media de tiempo de carga de la aplicación.

A pesar de ser una gran mejora se decidió mejorar más unificando la creación de los servicios propios de la aplicación móvil en uno, creando de ese modo una Factoría general que inicializara todos los servicios a la vez al iniciar la aplicación móvil, ya que antes se solicitaban y se llamaba los servicios cuándo se consideraba necesario, estando descentralizados, dificultando el mantenimiento de estos y lastrando la propia aplicación móvil.

Gracias a esta factoría se logró reducir un 10 % más el tiempo de carga de la aplicación móvil, logrando un tiempo de carga aproximado medio de 15 segundos.

Tarea 5 - Implantación de servicio de descarga de archivos. (MR-08)

Esta tarea viene derivada de la ejecución del requisito **AR-02**, ya que los archivos que superaban los 250MB no se enviaban en base64 en la petición, descargando ese archivo si fuera necesario, directamente desde WordPress.

Se implementó pues un servicio que descargara el archivo a partir de su "url" y lo almacenara en local en base 64 una vez descargado.

Lo primero que había que tener en cuenta es cómo saber qué archivo descargar, por lo que al solicitar el archivo el primer paso era buscar el id del "attachment" asociado. Tras buscar el id del "attachment" se extrajo de la base de datos el objeto con los datos de ese id.

Después se creó un método que tuviera como parámetro el objeto que hemos extraído de la base de datos. Antes de solicitarlo, había que tener en cuenta algo obvio, que el usuario debía estar conectado a internet. Tras esto se extrajo la "url" y se descargó el archivo solicitado de más de 250MB convirtiéndolo a base64 y almacenándolo en el registro de archivos de nuestra aplicación.

```
public bool DownloadFileBase(TAttach attach)
{
    if (FactoriaGeneral.getConnectivityOn().IsConnected)
    {
        string pdfUrl = attach.Url;
        try
        {
            using (WebClient client = new WebClient())
            {
                var bytes = client.DownloadData(pdfUrl);
                string base64String = Convert.ToBase64String(bytes);
                //esta bien hasta aqui
                FactoriaGeneral.getFileManager().writeFromBase64(attach.getFileName(), base64String);
                attach.Downloaded = 1;
                SQLite.SQLiteAsyncConnection _conn = FactoriaGeneral.getDataService();
                _conn.UpdateAsync(attach).Wait();
                return true;
            }
        }
        catch {
            this.ShowToast("Error en la solicitud, inténtelo de nuevo");
            return false;
        }
    }
    else {
        this.ShowToast("Se necesita conexion a internet para acceder a este elemento");
        return false;
    }
}
```

Ilustración 21 - Método de Descarga de archivo

Tarea 6 – Estandarización de pantallas de visualización/tipo. (MR-03)

Esta tarea viene derivada de la **Tarea 3** realizada en el **Sprint 1**. En ella se definieron para cada tipo de dato que opciones de visualización dispone. Como se ha indicado en la tarea anterior el objetivo es hacer que la aplicación se comporte como una página web, determinando que vistas por defecto se generan por cada tipo de dato.

Por lo que en esta tarea el objetivo es implementar en la aplicación móvil la estandarización de las pantallas definidas para cada tipo de dato.

Se creó un método para definir el comportamiento de abrir una lista, usándolo cuando fuera necesario. En este método, que recibía un tipo de dato, extrae el tipo de lista a emplear para ese elemento del parámetro de entrada, genera la lista.

Tras hacer esto se llama a otro método que decide que plantilla de detalle se aplicará para mostrar los datos en la lista, se extraen todos los datos de ese tipo de la base de datos o bien se extraen solo los datos solicitados por el filtro y se unen los datos a lista. Tras esto se muestra la lista.

```
protected Page getViewLista(TTipoContenido TTipo,bool bBind=true) {
    View aux;
    Page pag;
    String sTemplateList = TTipo.ventanaLista;
    String sTemplateElement = null;
    if (sTemplateList.Contains("#")){
        String[] sAux = sTemplateList.Split('#');
        sTemplateList = sAux[0];
        sTemplateElement = sAux[1];
    }
    aux = getViewListaByName(sTemplateList, sTemplateElement);
    if (TTipo.isFlag((long)TTipoContenido.FlagsComportamiento.muestraFiltrosDrawer)){
        pag = new BaseCPDrawer(){ ContentInt = aux, Content2 = new appGeneric.Views.ViewFiltros() };
    }else{
        pag = new ContentPageBase(){ContentInt = aux,};
    }if(bBind){
        pag.BindingContext = getBindingLista(TTipo);
        LaunchInitialitationAsync(pag.BindingContext);
    }
    LaunchInitialitationAsync(aux, pag.BindingContext);
    return pag;
}
```

Ilustración 22 - Método de lista

Se creó otro método para definir el comportamiento de abrir un detalle. Este método recibe un objeto "data", extrae el tipo del parámetro de entrada y busca que plantilla tiene ese tipo asociado para mostrar el detalle. Tras esto se llama a otro método que une los datos del elemento a la plantilla. Tras esto se muestra el detalle.

```
protected Page getViewDetail(TContenidos Tcon){
    TTipoContenido TTipo = Tcon.TTipo;
    appGeneric.ViewModels.ViewModelBase binding;
    View aux;
    switch (TTipo.ventanaDetalle){
        default:
            appGeneric.ViewModels.ItemVM miB= new appGeneric.ViewModels.ItemVM();
            miB.Item = Tcon;
            miB.Titulo = Tcon.ClasfTipo.Titulo;
            binding = miB;
            break;}
    try{
        String sAux = TTipo.ventanaDetalle;
        Type t = Type.GetType("appGeneric.Views.ListDetail.Detail."+ sAux);
        aux=(View) Activator.CreateInstance(t);
    }catch(Exception ex){
        aux = new appGeneric.Views.ListDetail.Detail.ItemDetailTarjeta();
    }
    LaunchInitialitationAsync(binding);
    LaunchInitialitationAsync(aux,binding);
    ContentPageBase pag = new ContentPageBase {ContentInt=aux, BindingContext=binding };
    return pag;
}
```

Ilustración 23 – Método de detalle

Por último, se creó una serie de métodos de consulta a la base de datos para saber qué tipos de datos tienen un "flag" activo en concreto, extrayendo los datos de los tipos de los que cumplan la condición.

Tarea 7 – Implantación de menús dinámicos. (MR-05)

Esta tarea viene derivada de la Tarea 4 del Sprint 1. En esa tarea se realizó la implantación de los menús dinámicos en lo que compete a la aplicación web, además se definió la estructura del "json" para enviar esos datos, por lo que en este punto el objetivo es hacer lo mismo con la parte móvil.

Se amplió el modelo de datos de la aplicación para que la aplicación pudiera trabajar con la configuración de menús realizada, creando de este modo la siguiente clase:

```
namespace appGeneric.Model.ConfigData
{
    [Xamarin.Forms.Internals.Preserve(AllMembers = true)]
    8 referencias
    public class  MenuConfig
    {
        5 referencias
        public enum Tipo : long
        {
            abrirTipo = 1,
            abrirTiposFlags = 2,
            abrirIDElemento = 3,
            abrirPagView=4,
            separador=5
        }
        public long tipo;
        public String titulo;
        public long idIcono;
        public long modifLong=0;
        public String modifString=null;
        public String modifString2 = null;
        public bool modifBool = false;
    }
}
```

Ilustración 24 - Extensión del modelo de datos

Después, se creó una clase en la que implementamos el método para extraer los datos del menú de la base de datos móvil y un método que a partir de los datos del menú genera una lista con objetos de menú para utilizar más adelante. Esta clase no se comportará como un servicio más por lo que no se gestionará desde "FactoriaGeneral" sino que será una extensión de la configuración de la aplicación.

```
public override string getConfig(string sNombre)
{
    String sLang = FactoriaGeneral.GetMultilingual().CodCurrentLanguageString;
    string cadena = sNombre + '_' + sLang;
    TContenidos Tcon = FactoriaGeneral.getQuicKDB().GetContenidoTitulo(cadena);
    return Tcon.Content;
}
```

Ilustración 25 - Método de solicitud del json del menú.

Tras esto, se creó una nueva consulta a la base de datos para solicitar por nombre su elemento data asociado. Esto se hizo para agilizar el proceso y poder buscar el elemento con los datos de menú sin tener que buscar en la base de datos de WordPress el "id" asociado a ese objeto para buscar internamente por "id".

Tras extraer el "json" de la base de datos y haber definido la clase de menú, se deserializó el "json" dando como resultado una lista de elementos de menú. Lo que había que hacer, entonces, es crear otro método que a partir del tipo de cada elemento de menú definiera su comportamiento. Para este propósito se creó una clase intermedia con los datos finales de menú:

```
public class MiMenu
{
    public ImageSource Icono;
    public String Titulo;
    public DataTemplate Destino2;
}
```

Ilustración 26 - Clase intermedia

En ella podemos ver que tiene los atributos **icono** y **titulo**, que definirán la visualización del elemento del menú y su comportamiento "**Destino2**" al clicar sobre él.

Ilustración 27 - Elemento de menú.

Por último, se asignó a esta lista de elementos de menú su vista correspondiente. (Será diferente la visualización del menú lateral que del menú superior)

Tarea 8 – Estandarización de estilos. (MR-04)

Para la estandarización de estilos se pensó cuáles de los estilos que usábamos en la aplicación podían definirse en la API. Tenemos dos tipos diferentes de estilos en la aplicación:

Estilos propios de la aplicación: Como por ejemplo el color de los elementos principales, color de texto por defecto, etc. Están definidos en código "xaml" y se utilizan antes incluso de hacer la petición de los datos a la API. Por lo tanto, por el momento, no es extraíble a la misma.

Estilos "css" para los navegadores nativos: Esto es debido a que hay tipos de datos que usan "html" para mostrar información y sin un fichero de estilos la información no se muestra de manera correcta. Estos estilos serán los que pasemos a gestionar a través de la API.

Gracias a la implementación de los menús dinámicos en la anterior tarea nos fue muy sencillo definir el fichero de estilos "css" ya que seguimos la misma filosofía que en ese caso. Tras definir el contenido del archivo en el gestor de contenidos y enviarlo a la aplicación móvil a través de la API, lo leemos de la base de datos móvil de misma manera que los menús, por nombre, y lo almacenamos localmente para su uso posterior.

Review

En este sprint, se realizaron los requisitos **MR-06, MR-02, MR-05, MR-04, MR-01, MR-03** y los requisitos **MR-08** y **MR-09** extraídos del Sprint 1.

No ha quedado ningún requisito sin hacer de los que se habían identificado para ser realizados en este Sprint y no ha aparecido ningún nuevo requisito para ser realizado.

En este Sprint gracias a la eliminación de dos requisitos en la Review del anterior Sprint se han conseguido terminar todos los requisitos planteados tardando un día menos de tiempo que el estimado para este Sprint.

Capítulo 5

Este capítulo se centrará en la realización de un prototipo funcional a modo de demostración para presentar sobre un ejemplo en concreto el trabajo realizado en el proyecto.

Sprint 3

Identificación

El requisito que se ha identificado, como se ha explicado al inicio del capítulo, para ser realizado en este sprint es el requisito **CSR-01**, Creación de prototipo. Ya se avanzó sobre él en el anterior "sprint" por necesidades del desarrollo. El objetivo en este sprint es conseguir una aplicación funcional en Android que refleje el trabajo realizado.

Proceso

Tarea 1 – Creación de Prototipo (CSR-01)

En este punto, se dividió en dos bloques el desarrollo, personalización de la aplicación móvil y la creación de vistas mínimas en la aplicación móvil.

Aunque ya se avanzó en la población de la base de datos de WordPress en el Sprint anterior (Ver Tarea 2, Sprint 2), en este punto mencionaremos el proceso que se siguió para realizar el prototipo. Lo primero que se hizo fue definir el propósito de la aplicación, se determinó que la aplicación móvil estaría enfocada en la Universidad de La Rioja, siendo capaz de mostrar noticias, eventos, avisos y e información de interés como teléfonos de contacto etc.

Tras esto, se exportó la arquitectura del modelo de datos del proyecto para el Ayuntamiento de Ezcaray, eliminando los tipos de datos previamente a la importación que no se necesitaban para nuestro propósito. Después, se asignó a cada tipo de dato su tipo de comportamiento y su asignación de vistas correspondientes y se definieron los menús a mostrar y el fichero de estilos.

También se buscó información relativa a la Universidad de La Rioja, como la comentada anteriormente, poblando la base de datos de noticias, eventos, avisos, vistas de teléfonos de contacto, de localizaciones de interés, etc. Tras esto se comenzó la personalización propia de la aplicación móvil.

El primer paso es personalizar la visualización de la pantalla de carga, se eligió un gif de carga y un gif personalizado.

Tras esto, se personalizó el menú lateral y superior de la pantalla de inicio, diseñando la imagen del fondo del menú, incluyendo la imagen superior con el logo de la UR y los logos de las opciones de menú.

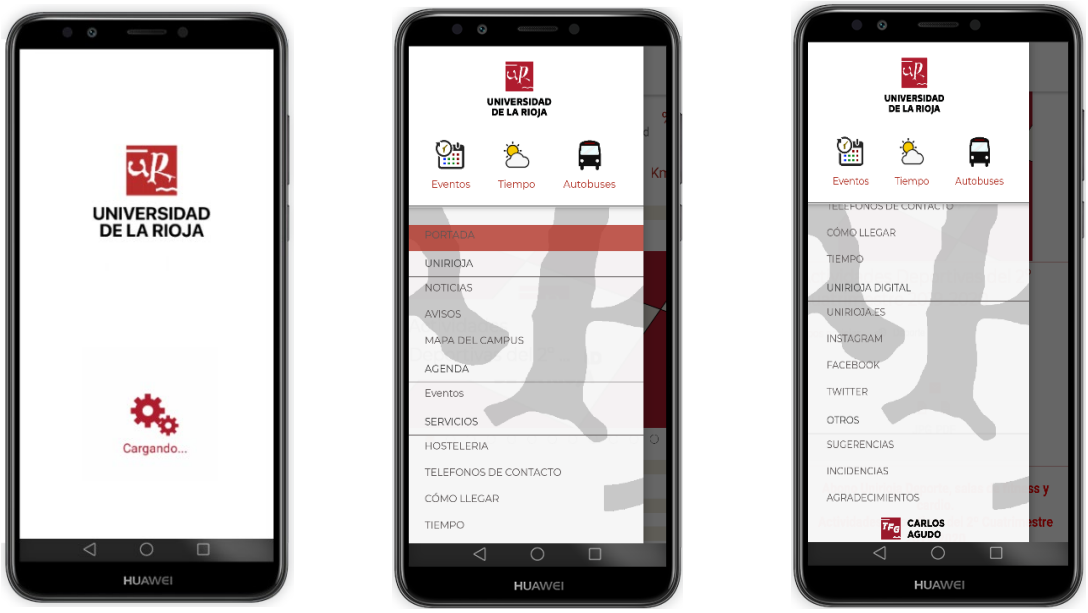


Ilustración 28 - Pantalla de carga aplicación móvil y menú lateral y superior.

Posteriormente se diseñó y estableció el icono personalizado de la aplicación que se mostrará en el contenedor de aplicaciones y cuando se contrae la aplicación y el nombre de la aplicación móvil.

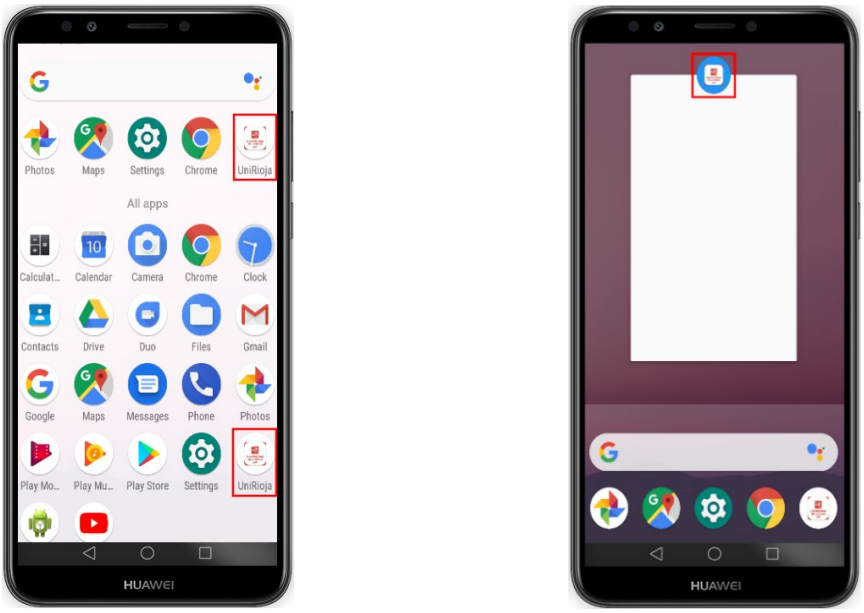


Ilustración 29 - Personalización de iconos.

En cuanto a la personalización de ventanas en la plataforma web se diseñaron y definió una serie de visualizaciones que se utilizaban en la aplicación móvil. Un ejemplo de esto son por ejemplo la visualización de los teléfonos de contacto o el formulario de incidencias.

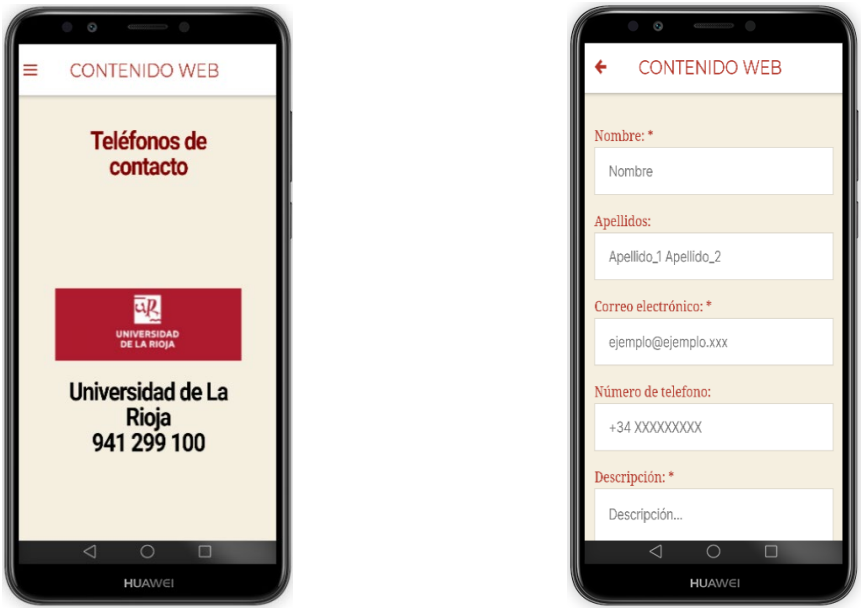


Ilustración 30 – Visualizaciones de la aplicación móvil

Por último, se añadieron una serie de etiquetas e imágenes para terminar de perfeccionar el prototipo.

Ilustración 31 - Vista del prototipo

Review

En este sprint, se realizó el requisito **CSR-01**. No ha quedado ningún requisito sin hacer de los que se habían identificado para ser realizados en este Sprint y no ha aparecido ningún nuevo requisito para ser realizado, por lo que una vez acabado este sprint se da por finalizado la parte de desarrollo de este trabajo.

Como reflexión, destacar que a pesar de que el trabajo de construir la aplicación nos costó dos días más de lo esperado, dedicando en total una semana de trabajo a este propósito, gracias a las mejoras realizadas en el presente Trabajo de Fin de Grado, haber partido desde el mismo inicio adaptando un nuevo concepto de aplicación diferente del concepto inicial de la aplicación del Ayuntamiento de Ezcaray hubiera costado como mínimo tres veces más, ya que solo la creación de cero todo el modelo de datos, de menús y la creación de vistas para cada tipo de dato, perfectamente costaría esas tres veces más.

Capítulo 6

Seguimiento y control

En este punto analizaremos las desviaciones en cuanto a tiempo respecto a la planificación que hicimos inicialmente.

Duración real

PLANIFICACIÓN DE TIEMPOS				
Tarea		**Horas Estimadas**	**Horas Reales**	**Desviación**
Sprint 0	Análisis	20 horas	20 horas	0%
	CR-02	5 horas	5 horas	0%
	CR-04	3 horas	3 horas	0%
	AR-01	2 horas	2 horas	0%
	CR-03	5 horas	5 horas	0%
	CR-01	5 horas	10 horas	**100%**
Total:		**40 horas**	**45 horas**	13%
Sprint 1	AR-03	10 horas	15 horas	**50%**
	AR-02	20 horas	20 horas	0%
	AR-06	20 horas	25 horas	25%
	AR-04	20 horas	20 horas	0%
	AR-07	0 horas	5 horas	–
	AR-05	20 horas	0 horas	–
Total:		**90 horas**	**85 horas**	-6%
Sprint 2	MR-07	0 horas	10 horas	–
	MR-02	15 horas	15 horas	0%
	MR-01	10 horas	5 horas	**-50%**
	MR-08	0 horas	10 horas	–
	MR-03	30 horas	30 horas	0%
	MR-05	25 horas	25 horas	0%
	MR-04	5 horas	5 horas	0%
	MR-06	20 horas	0 horas	–
Total:		**105 horas**	**100 horas**	-5%
Sprint 3	CSR-01	15 horas	25 horas	**67%**
Total:		**15 horas**	**25 horas**	**67%**
	Memoria y Anexos	30 horas	30 horas	0%
	Reuniones	5 horas	5 horas	0%
	Seguimiento y Control	15 horas	15 horas	0%
Total:		**50 horas**	**50 horas**	0%
TOTAL:		**300 horas**	**300 horas**	0%

Tabla 13 – Duración Real

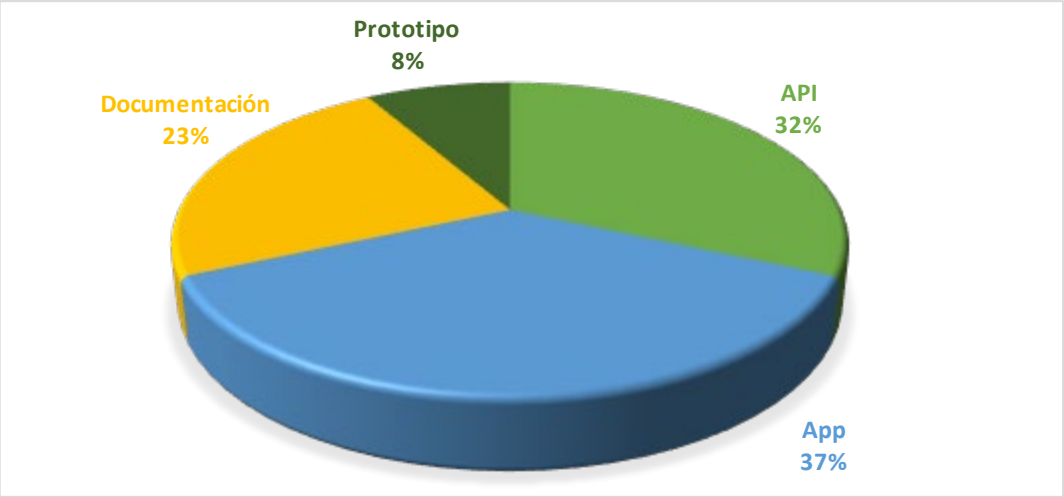


Ilustración 32 - Gráfico de sectores de la carga de trabajo real

Desviaciones

Podemos observar una serie de desviaciones importantes de la planificación, la mayoría derivadas por una mala planificación, al no entender, en un principio, la envergadura del proyecto en sí mismo y el trabajo que suponía realizar ciertas tareas.

Las tareas que más desviación han tenido son, en orden descendente:

- **CR-01 Limpieza de código:** Esta tarea ha tenido una desviación del **100%**, ya que se planificó que iba a ser realizada en un día (5 horas), pero finalmente hubo que destinar dos días. Esto fue debido a debido a que, como se comentó en el Sprint 0, no fuimos realmente conscientes de todo aquello que había que eliminar hasta que no estuvimos realmente frente al problema, ya que no solo se eliminó código, sino que también hubo que eliminar imágenes, clases inútiles, y fuentes.

- **CSR-01 Ejemplo de creación de aplicación móvil sobre la Universidad de La Rioja a partir del paquete:** Esta tarea estaba planeada para ser realizada en 15 horas, pero finalmente nos tomó 25, resultando una desviación del **67%**, ya que, se tuvo que realizar además de la personalización de la aplicación móvil y la población de la base de datos, el diseño de todas las imágenes que personalizan la aplicación.

- **AR-03 Definir estándar para la estructura de datos de la API:** Tarea con una desviación del **50%**, planeada a ser realizada en 10 horas pero que finalmente fue realizada en 15, un día más de lo esperado. Esto fue debido a que surgieron nuevas ideas a realizar durante el desarrollo.

- **MR-01 Reducción tiempo de carga de la aplicación:** Este caso es diferente ya que se tardó menos tiempo del que se había planteado, pasando de 10 horas planificadas a 5 horas reales, con una desviación del **-50%.** Esto es debido a que este requisito se benefició de los avances realizados en otros requisitos.

Diagrama Gantt Tiempos Reales

Diagrama Gantt

Paquetes de trabajo		Noviembre				Diciembre				Enero				Febrero	
		S1 25h	S2 25h	S3 25h	S4 25h	S1 15h	S2 20h	S3 25h	S4 0h	S1 15h	S2 25h	S3 25h	S4 25h	S1 25h	S2 25h
Sprint 0	Análisis	20h													
		20h													
	CR-02	5h													
		5h													
	CR-04		3h												
			3h												
	AR-01		2h												
			2h												
	CR-03		5h												
			5h												
	CR-01		5h												
			10h												
Sprint 1	AR-03		10h												
			5h	10h											
	AR-02			20h											
				15h	5h										
	AR-06			5h	15h										
					20h	5h									
	AR-04				10h	10h									
						10h	5h								
	AR-07														
							5h								
	AR-05					5h	15h								
Sprint 2	MR-07														
							5h	5h							
	MR-02							15h							
								15h							
	MR-01							10h							
								5h							
	MR-08														
										10h					
	MR-03									15h	15h				
										5h	25h				
	MR-05										10h	15h			
												25h			
	MR-04											5h			
													5h		
	MR-07											5h	15h		
Sprint 3	CSR-01						5h						10h		
							5h						20h		
Tareas Transversales	Memoria														
	Reuniones														
	Seguimiento														

Tabla 14 – Duración Real

Tiempo Estimado
Tiempo Real
Requisito no realizado
Requisito no contemplado en el análisis
Semana no laboral

Riesgos materializados

Fuente	Riesgo	Causa	Solución
Metodología	Durante el desarrollo pueden aparecer requisitos no contemplados	Debido a que no se pudieron definir con precisión todos los requisitos de manera previa al desarrollo.	Como solución, para no excedernos en el número de horas permitidas se decidió que los requisitos determinados en el punto siguiente (Exclusiones finales) no entrarían en este trabajo, pero se plantearían como futuras mejoras.

Tabla 13 - Riesgos Materializados

Exclusiones finales

No se han realizado los requisitos siguientes por exceso de tiempo ya que han surgido otros requisitos a los que hemos dotado de más prioridad. Estos requisitos pasarán a formar parte del trabajo futuro. Eliminar estos dos requisitos nos ha brindado de 40 horas para poder realizar otros requisitos de nueva aparición.

Código	Titulo
AR-05	Implantación de multidioma.
MR-07	Implantación de multidioma.

Tabla 15 - Tabla de Exclusiones Finales

Capítulo 7

Conclusiones

Objetivos alcanzados

El objetivo de este trabajo era abstraer una aplicación móvil que pudiera servir de carcasa vacía para que según los datos que se le introduzcan se convierta en una aplicación o en otra, es decir, que el propósito de la aplicación móvil dependa de los datos que se le introduzcan.

Esto es algo que hemos conseguido realizar en este proyecto, ya que se han completado la mayoría de los requisitos impuestos en la fase de análisis, dejando sin hacer los requisitos comentados en la sección de Exclusiones Finales.

Además, se han generado los documentos solicitados para facilitar el proceso de creación y configuración de la aplicación web y móvil para el uso interno de la empresa en un futuro.

Gracias a este trabajo se han sentado las bases para la empresa para seguir añadiendo funcionalidad tanto a la aplicación web, como a la aplicación móvil, pero teniendo un producto final para poder mostrar un prototipo, de la misma manera que se ha hecho en este proyecto, y así atraer nuevos municipios a adquirir el producto. Se ha logrado identificar el camino por recorrer de la empresa en relación con la generación y el desarrollo de aplicaciones móviles.

Hacer notar que pese a todo el trabajo realizado en este proyecto todavía queda mucha funcionalidad que seguir implementando y mejorando.

Resaltar la conformidad notable de mi tutor de TFG en la empresa, como representante de la misma, del producto finalmente conseguido mediante la realización del presente trabajo.

Conclusiones personales

Lo primero que me gustaría destacar es que al inicio del proyecto tenía serias dudas acerca de la metodología de trabajo elegida. Decidí usar una metodología ágil iterativa debido a la posibilidad de que surgieran nuevos requisitos en el trascurso del proyecto, y el propio trabajo me ha dado la razón en este aspecto. Si no hubiera elegido una metodología ágil, los requisitos nuevos que han surgido, no se hubieran podido realizar y, por tanto, hubiera sido perjudicial para el resultado último de este proyecto ya que estos nuevos requisitos no contemplados eran necesarios.

También me gustaría destacar y agradecer la comunicación con mi tutor de prácticas en la empresa, que siempre me ha estado apoyando en todas las fases del desarrollo, quitándole importancia a problemas surgidos. Problemas que para mí en un principio parecían de más envergadura de la que en realidad tenían, aprendiendo gracias a este proceso a ser más optimista en cuanto a mis conocimientos adquiridos, porque pese a mi reciente incorporación al mundo laboral, tengo más conocimiento del que me presuponía inicialmente por mi falta de experiencia en este mundo.

También me ha ayudado a ser más independiente, dándome alas cuando él lo consideraba, dejándome acertar, fallar y aprender de mis errores. Aunque lo más importante que me gustaría destacar es el apoyo constante que ha depositado en mí, animándome y alentándome, demostrándome que realmente era capaz de resolver lo que se había planteado en un principio para el proyecto.

Agradecer también a mi tutor de la Universidad de La Rioja, Jesús María Aransay Azofra, que me ha dado un apoyo constante durante todo el tiempo que he estado en la empresa (ya que también ha sido mi tutor de prácticas), abriéndome su puerta siempre que se lo solicitaba, animándome en los momentos de más agobio y liberando de mí en muchos casos la presión constante autoimpuesta a la que me veía sometido.

En este proyecto, además, he entendido de primera mano, lo difícil y necesario que es realizar una buena planificación, determinando de manera precisa los requisitos del proyecto, no intentando abarcar más de lo que verdaderamente es posible realizar.

Trabajo futuro

Se han identificado posibles mejoras para continuar añadiendo funcionalidad:

- **Implementación de multidioma:** Mejora que viene heredada de este trabajo ya que por motivo de tiempo no me ha sido posible poder realizarla. El objetivo es hacer que el lenguaje se adecue a las necesidades de cada usuario en función de en qué lenguaje esté navegando por la página web o en función de en qué idioma tiene configurado el dispositivo móvil.

- **Implementación de gestión de usuarios:** Mejora de la aplicación móvil para que cada usuario pueda tener un espacio personal dentro de la aplicación.

- **Implementación de servicios "push":** Mejora que consiste en la implantación de un servicio "push" que avise a todos los usuarios, de cualquier información relativa a un aviso al que se hayan suscrito desde la aplicación móvil.

- **Implementación de servicios "pop-up":** Mejora que consiste en la implantación de un servicio "pop-up" que avise a todos los usuarios, estén registrados o no, de futuros eventos, publicidad, etc.

Bibliografía

Xamarin:

https://forums.xamarin.com/

https://docs.microsoft.com/es-es/xamarin/xamarin-forms/

WordPress:

https://developer.wordpress.org/

PHP:

https://phptherightway.com/

XAMPP:

https://www.apachefriends.org/es/index.html

Brackets:

http://brackets.io/

Visual Studio:

https://visualstudio.microsoft.com/support/

Stack Overflow:

https://es.stackoverflow.com/

Anexos

Anexo I - Seguimiento de reuniones

Acta de Reunión 1	
Fecha:	04-11-2019
Asistentes:	Jesús Navajas Briones Rubén Llamas Ramos Carlos Agudo Postigo
Hora inicio:	13:00
Hora finalización:	13:30
Orden del día:	Análisis del sistema. Definición de requisitos.

Conclusiones:
Se determinaron los requisitos a realizar en base al análisis realizado, así como el alcance, exclusiones y entregables.

Acta de Reunión 2	
Fecha:	18-11-2019
Asistentes:	Jesús Navajas Briones Carlos Agudo Postigo
Hora inicio:	13:00
Hora finalización:	13:30
Orden del día:	Cerrar Sprint 0. Plantear Sprint 1.

Conclusiones:
Se validaron los requisitos realizados en el Sprint 0 y se discutió acerca de los requisitos que se iban a realizar en el Sprint 1.

Acta de Reunión 3	
Fecha:	02-12-2019
Asistentes:	Jesús Navajas Briones Carlos Agudo Postigo
Hora inicio:	13:00
Hora finalización:	13:30
Orden del día:	Cerrar Sprint 1. Plantear Sprint 2.

Conclusiones:
Se validaron los requisitos realizados en el Sprint 1 y se discutió acerca de los requisitos que se iban a realizar en el Sprint 2.

Acta de Reunión 4	
Fecha:	08-01-2020
Asistentes:	Jesús Navajas Briones Carlos Agudo Postigo
Hora inicio:	13:00
Hora finalización:	13:30
Orden del día:	Control Sprint 1 Post vacacional.

Conclusiones:

Reunión meramente informativa de control donde se identificaron qué requisitos se habían hecho, qué requisitos se estaban haciendo y qué requisitos quedaban por hacer en el Sprint 1.

Acta de Reunión 5	
Fecha:	27-01-2020
Asistentes:	Jesús Navajas Briones Carlos Agudo Postigo
Hora inicio:	13:00
Hora finalización:	13:30
Orden del día:	Cerrar Sprint 2. Plantear Sprint 3.

Conclusiones:

Se validaron los requisitos realizados en el Sprint 2 y se discutió acerca de los requisitos que se iban a realizar en el Sprint 3.

Acta de Reunión 6	
Fecha:	10-02-2020
Asistentes:	Jesús Navajas Briones Carlos Agudo Postigo
Hora inicio:	13:00
Hora finalización:	13:30
Orden del día:	Reunión fin de proyecto. Revisión de objetivos.

Conclusiones:

Se validaron el requisito realizados en el Sprint 3 y se cerró la parte de implementación, valorando y evaluando los objetivos realizados y los objetivos que quedan por hacer en el futuro.